|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное автономное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ: Информатики и системы управления (ИУ)

КАФЕДРА: Информационные системы и телекоммуникации (ИУ3)

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

**\_\_\_\_\_\_\_\_\_\_\_Информационная\_система\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ресторона\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_**

**\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

**\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент ИУ3-53Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** А.Р. Измайлов

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** И. Г. Боровик

(Подпись, дата) (И.О.Фамилия)

*2025 г.*

# АННОТАЦИЯ

Пояснительная записка содержит описание полного цикла исследования и разработки системы и программного комплекса, предназначенного для автоматизации деятельности ресторана. Ключевой задачей работы является повышение эффективности обслуживания гостей и оптимизация складского учета за счет использования современных информационных решений.

В работе представлен детальный анализ бизнес-процессов ресторана, выделены основные потоки данных и роли пользователей. На этапе проектирования сформированы требования к функционалу системы, разработаны диаграммы вариантов использования и архитектурные схемы взаимодействия клиентских приложений с сервером. Особое внимание уделено построению реляционной модели данных, нормализованной до четвёртой нормальной формы для обеспечения целостности информации, а также для удобства работы с ней.

Практическая часть работы включает реализацию серверного API на базе платформы Java Spring Boot и двух клиентских веб-интерфейсов (для посетителей и персонала) с использованием библиотеки React. Описана логика работы основных модулей: бронирования столиков, управления меню, обработки заказов и учета товарных запасов. Подведены итоги разработки, подтверждающие работоспособность системы и соответствие поставленным техническим требованиям.

СОДЕРЖАНИЕ

[АННОТАЦИЯ 2](#_Toc216141217)

[ВВЕДЕНИЕ 6](#_Toc216141218)

[Актуальность темы 6](#_Toc216141219)

[Цель работы 7](#_Toc216141220)

[Научная новизна 9](#_Toc216141221)

[Практическая значимость 10](#_Toc216141222)

[ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ 11](#_Toc216141223)

[1. Подготовка 11](#_Toc216141224)

[1.1. Описание предметной области 11](#_Toc216141225)

[1.2. Функциональная модель бизнес-процессов 14](#_Toc216141226)

[1.3. Глоссарий предметной области 24](#_Toc216141227)

[1.4 Документ “Видение” 26](#_Toc216141228)

[2. Анализ 29](#_Toc216141229)

[2.1 Концептуальная модель предметной области 29](#_Toc216141230)

[2.2. Функциональные требования 30](#_Toc216141231)

[2.3. Диаграммы вариантов использования 34](#_Toc216141232)

[2.4 Краткое и подробное описание вариантов использования 36](#_Toc216141233)

[2.5. Проектирование пользовательского интерфейса 43](#_Toc216141234)

[3. Проектирование 48](#_Toc216141235)

[3.1. Диаграммы анализа 48](#_Toc216141236)

[3.2. Диаграммы последовательности 50](#_Toc216141237)

[3.3. Диаграммы классов этапа проектирования 54](#_Toc216141238)

[3.4. Схема (структура) базы данных 56](#_Toc216141239)

[4. Создание 60](#_Toc216141240)

[4.1. Выбор архитектуры программного обеспечения 60](#_Toc216141241)

[4.2. Используемые инструменты и технологии веб-приложения 62](#_Toc216141242)

[4.3. Создание базы данных 62](#_Toc216141243)

[4.4. Создание программы системы 64](#_Toc216141244)

[4.5. Тестирование 65](#_Toc216141245)

[КОНСТРУКТОРСКАЯ ЧАСТЬ 67](#_Toc216141246)

[1. Техническое задание 67](#_Toc216141247)

[1.2. Введение 68](#_Toc216141248)

[1.3. Основание для разработки 68](#_Toc216141249)

[1.4. Требования к функциональным характеристикам 68](#_Toc216141250)

[1.5. Требования к надежности 69](#_Toc216141251)

[1.6. Условия эксплуатации 69](#_Toc216141252)

[1.7. Требования к составу и параметрам технических средств 70](#_Toc216141253)

[1.8. Требования к информационной и программной совместимости 70](#_Toc216141254)

[1.9. Требования к маркировке и упаковке 71](#_Toc216141255)

[1.10. Требования к транспортировке и хранению 71](#_Toc216141256)

[1.11. Требования к программной документации 72](#_Toc216141257)

[1.12. Технико-экономические показатели 73](#_Toc216141258)

[1.13. Стадии и этапы разработки 73](#_Toc216141259)

[2. Архитектура системы 75](#_Toc216141260)

[3. База данных 77](#_Toc216141261)

[3.1. Обоснование выбора СУБД 77](#_Toc216141262)

[3.2. Схема базы данных и описание таблиц 77](#_Toc216141263)

[4. Серверная часть системы 80](#_Toc216141264)

[4.1. Обоснование средств разработки 80](#_Toc216141265)

[4.2. Описание схемы бизнес-логики 80](#_Toc216141266)

[4.3. Спецификация модулей 81](#_Toc216141267)

[5. Интерфейс пользователя 84](#_Toc216141268)

[5.1. Клиентское приложение 84](#_Toc216141269)

[5.2. Приложение для персонала 85](#_Toc216141270)

[ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 88](#_Toc216141271)

[1. Анализ исходных данных 88](#_Toc216141272)

[2. Тестирующие драйверы и заглушки 90](#_Toc216141273)

[Тестирующий драйвер (Driver) 90](#_Toc216141274)

[Тестирующие заглушки 91](#_Toc216141275)

[3. Протокол тестирования 92](#_Toc216141276)

[4. Исследование системы на производительность и отказоустойчивость 93](#_Toc216141277)

[ЗАКЛЮЧЕНИЕ 94](#_Toc216141278)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 96](#_Toc216141279)

# ВВЕДЕНИЕ

В данном разделе объясняется выбор темы работы, рассматриваются проблемы, с которыми сталкиваются рестораны при ручном управлении, и формулируются цели разработки новой системы. Также здесь описано, чем предлагаемое решение отличается от существующих аналогов и какую конкретную пользу оно принесет бизнесу.

Современный ресторан – это сложный механизм, где скорость и точность решают всё. Когда официанты, повара и администраторы работают разрозненно, используя бумажные блокноты и устные договоренности, неизбежно возникают ошибки. Заказы теряются, гости долго ждут счета, а на складе внезапно заканчиваются нужные продукты. В условиях высокой конкуренции клиенты просто уходят туда, где их обслуживают быстрее и качественнее. Поэтому старые методы работы уже не справляются с нагрузкой и мешают развитию заведения.

Внедрение информационной системы помогает навести порядок в этих процессах. Программа берет на себя всю рутину: мгновенную передачу заказа на кухню, автоматический подсчет стоимости и списание продуктов со склада. Это делает работу ресторана прозрачной и предсказуемой. Руководитель получает точные цифры вместо примерных оценок, а персонал тратит меньше времени на заполнение бумаг. Разработка собственной системы позволяет создать инструмент, идеально подходящий под конкретные нужды сети, без лишних сложных функций и переплат за стороннее ПО.

## Актуальность темы

Современный ресторанный бизнес – это сфера, где скорость обслуживания и качество учета напрямую влияют на прибыль. Во многих заведениях управление процессами до сих пор ведется по старинке: заказы записываются в блокноты, отчеты по продажам сводятся вручную в конце смены, а учет продуктов на складе ведется в бумажных журналах. Такой подход неизбежно приводит к проблемам.

Во-первых, это «человеческий фактор»: официант может забыть передать заказ на кухню или перепутать блюдо, а повар – не увидеть комментарий клиента. Во-вторых, отсутствие оперативной информации о запасах приводит к тому, что популярные блюда внезапно попадают в «стоп-лист» в самый разгар вечера, или, наоборот, продукты портятся из-за чрезмерных закупок. В-третьих, руководство не видит реальной картины бизнеса в режиме онлайн и получает отчеты с задержкой, когда исправлять ситуацию уже поздно.

Важным аспектом является и конкурентоспособность. В условиях насыщенного рынка общественного питания клиенты выбирают заведения не только по качеству кухни, но и по уровню сервиса. Долгое ожидание счета, ошибки в чеке или невозможность забронировать столик онлайн могут стать причиной оттока посетителей к более технологичным конкурентам. Информационные технологии сегодня становятся не просто преимуществом, а необходимым условием выживания бизнеса. Автоматизация позволяет внедрять программы лояльности, анализировать предпочтения гостей и делать персональные предложения, что невозможно реализовать при ручном учете.

Внедрение информационной системы позволяет решить эти проблемы комплексно. Переход на цифровой учет связывает зал, кухню и склад в единый механизм. Это делает процесс обслуживания прозрачным, ускоряет передачу информации и позволяет исключить ошибки, связанные с ручным вводом данных. Поэтому тема разработки такой системы является востребованной и актуальной.

## Цель работы

Главной целью работы является проектирование и программная реализация информационной системы, которая автоматизирует основные бизнес-процессы сети ресторанов. Система должна объединить разрозненные операции (бронирование, заказ, приготовление, списание продуктов) в единый цифровой контур.

Разработка собственной информационной системы, в отличие от использования готовых коробочных решений, позволяет учесть уникальные особенности бизнес-процессов конкретной сети ресторанов. Готовые продукты часто перегружены избыточным функционалом, сложны в настройке и требуют дорогостоящей подписки. Создание специализированного решения позволяет получить легкий, быстрый и полностью контролируемый инструмент, который можно гибко дорабатывать и масштабировать по мере роста бизнеса без лицензионных ограничений.

Для достижения этой цели необходимо применить объектно-ориентированный подход к проектированию, что позволит создать гибкую и масштабируемую архитектуру. Система должна гарантировать надежное хранение данных, их целостность (чтобы нельзя было удалить продукт, который используется в активном заказе) и предоставлять удобные интерфейсы для разных групп пользователей: от клиентов до администраторов.

**Задачи работы**

Для достижения поставленной цели необходимо поэтапно решить следующие задачи:

1. **Провести детальный анализ предметной области:**

* Изучить, как работают процессы обслуживания гостей и движения товаров в ресторане сейчас («как есть»).
* Найти проблемные места, где чаще всего возникают ошибки или задержки (например, потеря заказов или расхождения в инвентаризации).

1. **Выполнить проектирование системы:**

* Разработать визуальные модели системы с помощью языка UML (диаграммы вариантов использования, классов, последовательности) для четкого понимания логики работы [1, 2].
* Спроектировать схему базы данных и привести её к четвертой нормальной форме (4NF), чтобы исключить дублирование информации и обеспечить корректные связи между таблицами [3].

1. **Реализовать программный комплекс:**

* Написать программный код серверной части (API) для обработки данных и расчетов.
* Создать клиентские веб-интерфейсы для удобной работы персонала и посетителей.
* Реализовать алгоритмы проверки наличия продуктов на складе перед оформлением заказа.

1. **Провести тестирование и отладку системы:**

* Разработать сценарии тестирования для проверки всех функциональных требований.
* Проверить устойчивость системы к некорректным действиям пользователей и пиковым нагрузкам.
* Убедиться в корректности работы интерфейса на различных устройствах (компьютерах, планшетах).

## Научная новизна

В работе предложен подход к организации данных, который адаптирован именно под сетевую структуру ресторанов. В отличие от простых систем учета, разработанное решение включает:

* Механизм централизованного управления меню, позволяющий обновлять список блюд и цены одновременно во всех филиалах сети.
* Модуль аналитики, который позволяет строить прогнозы спроса на основе истории заказов, помогая заранее планировать закупки продуктов.

## Практическая значимость

Результаты работы имеют прямую практическую пользу для всех участников процесса:

* Для руководства: появляется инструмент контроля бизнеса в реальном времени и прозрачная отчетность.
* Для персонала: упрощается работа с заказами, исчезает необходимость вести бумажные журналы, снижается риск ошибок при расчетах.
* Для клиентов: повышается скорость обслуживания, а возможность онлайн-бронирования и актуальное меню делают посещение ресторана более комфортным.

В итоге, разработанная система служит фундаментом для цифровой трансформации ресторана, позволяя бизнесу расти и открывать новые филиалы без потери управляемости. Экономический эффект от внедрения системы достигается за счет сокращения прямых потерь продуктов (благодаря точному учету сроков годности и остатков) и увеличения оборачиваемости столиков (за счет ускорения обслуживания). Кроме того, цифровая база данных клиентов позволяет проводить маркетинговые кампании с высокой конверсией, возвращая гостей в ресторан снова и снова.

# ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

В данном разделе проводится анализ предметной области, рассматриваются бизнес-процессы ресторана и формируются требования к разрабатываемой информационной системе.

Результаты данного этапа являются фундаментом для всего проекта. Глубокое понимание предметной области позволяет избежать критических ошибок проектирования, которые на поздних стадиях разработки могут потребовать полной переделки архитектуры системы.

## 1. Подготовка

### 1.1. Описание предметной области

Чтобы точнее понять предметную область информационной системы ресторана, рассмотрим, в чём состоит работа ресторана в этой сети.



*Рисунок 1 – Предметная область*

**Клиент**, желающий пообедать, обращается к **официанту** или к **администратору**. В соответствии с пожеланиями клиента **официант** предлагает **столик**, а после записывает **заказ**. Затем талон заказа передаётся на кухню, где **повар** по бумажному каталогу меню собирает список ингредиентов и готовит выбранные **блюда**. После официант составляет счёт и клиент его оплачивает.

При этом используются следующие бумажные документы:

* **Журнал бронирований** *–* реестр заявок на **резерв столиков**, куда **администратор** вносит дату, время и количество гостей;
* **Журнал заказов** *–* карточки или бланки с подробностями каждого **заказа** (названия **блюд**, модификации, особенности);
* **Складской журнал** *–* записи об остатках **продуктов** и **сырья** в каждом **ресторане**;
* **Реестр поставок** *–* бумажные накладные от **поставщиков** с датой, наименованием и количеством привезённых **продуктов**;
* **Журнал продаж** *–* ежедневные сводки по **выручке**, количеству обслуженных **клиентов** и выданных **чеков**;
* **Реестр претензий** *–* бумаги с **отзывами**, **жалобами** и **предложениями клиентов**.

Особенности функционирования сети ресторанов:

* Каждый ресторан ведёт собственные бумажные журналы, централизованного обмена данными нет;
* Меню может отличаться для разных филиалов, но фиксируется в отдельных **каталогах блюд**;
* Запасы **продуктов** учитываются вручную, часто возникают ошибки и рассинхронизация;
* Нет оперативной статистики по популярности блюд, загруженности кухни, пиковым часам.

В настоящее время вся работа **администратора** и **повара** осуществляется без использования компьютеров. **Официант** вручную фиксирует **заказы**, **складской учёт** ведётся на бумаге, а менеджмент не располагает централизованным отчётом по продажам и остаткам.

Если проводить автоматизацию работы сети ресторанов, то в первую очередь хотелось бы автоматизировать следующие виды работ:

* Ведение **электронной базы бронирований** вместо **журнала бронирований**;
* **Регистрацию заказов** в единой системе вместо **журнала заказов**;
* Учёт **остатков продуктов** автоматически с интеграцией **склада** вместо **складского журнала**;
* Приём и обработку **накладных поставщиков** в электронной форме вместо **реестра поставок**;
* Формирование ежедневных **отчётов по выручке** и **загруженности кухни** вместо **журнала продаж**;
* Сбор и хранение **отзывов клиентов** в базе вместо **реестра претензий**;
* Аналитику популярности блюд, эффективности сотрудников и пиковых часов работы.

В проекте сеть ресторанов выступает как единая организация, где филиалы могут принадлежать головной компании или работать по франшизе. **Поставщики** обеспечивают регулярные **поставки**, **администраторы** управляют **бронированиями** и **продажами**, **повара** готовят **блюда**, а **официанты** обслуживают **клиентов**. Автоматизация позволит централизовать бумажный документооборот, повысить точность данных и оперативность управления сетью.

### 1.2. Функциональная модель бизнес-процессов

Для анализа предметной области функционирования ресторана и выявления процессов, подлежащих автоматизации, построена функциональная модель бизнес-процесса «Обработка заказа на блюдо». Моделирование выполнено по методологии IDEF0, позволяющей отразить структуру деятельности с выделением входов, выходов, управляющих воздействий и механизмов исполнения.

#### Контекстная диаграмма A0 «Обработка заказа на блюдо»

Контекстная диаграмма представляет собой верхнеуровневое описание процесса оформления заказа клиентом ресторана в существующей (неавтоматизированной) системе. Все действия выполняются вручную, с применением бумажных форм и журналов.

**Идентификатор:** A0

**Входы** – информация, подвергаемая обработке:

* **«Устное обращение клиента»** – начальная инициатива клиента, выраженная в форме устного запроса к официанту или администратору с целью получения блюда или набора блюд из меню ресторана.
* **«Меню ресторана» (бумажный каталог блюд)** – справочный документ, предоставляемый клиенту, в котором содержатся наименования блюд, их описание, стоимость и принадлежность к определённой категории (горячее, салаты и т.д.).
* **«Денежные средства»** – оплата, передаваемая клиентом в обмен на выбранное блюдо; могут быть представлены в виде наличных или других форм расчёта (в текущей системе – наличными).

**Выходы** – данные, полученные в результате выполнения работы:

* **«Блюдо»** – результат кулинарной обработки, выполненной поваром на основании заказа клиента и рецептуры, представленное к подаче на стол.
* **«Квитанция об оплате»** – подтверждающий документ (или устное подтверждение), выдаваемый клиенту после завершения расчёта; фиксирует сумму, дату и предмет оплаты.
* **«Добавление в журнал заказов»** – регистрация информации о заказе в бумажном журнале ресторана; включает дату, наименование блюда, обслуживающего сотрудника и сумму.

**Управляющие воздействия** – условия, ограничения, указания:

* **«Стандарты обслуживания клиентов»** – регламентирующие документы, определяющие правила взаимодействия персонала с клиентами, допустимое время ожидания, нормы вежливости и качества обслуживания.
* **«Каталог блюд»** – внутренний документ, описывающий состав, рецептуру, стоимость и технико-технологические параметры блюд. Используется для сверки состава и возможности приготовления.
* **«График работы поваров»** – документ, содержащий информацию о сменах и доступности кулинарного персонала на кухне, необходимый для корректной маршрутизации заказов.
* **«Технические карты блюд»** – технологическая документация, определяющая стандарты приготовления каждого блюда: ингредиенты, объём, последовательность, термическая обработка.
* **«Складской журнал остатков ингредиентов»** – бумажный документ, отражающий наличие продуктов и сырья в конкретном ресторане. Используется для принятия решения о возможности приготовления блюда.

**Механизмы или исполнители** – участники процесса, выполняющие работу:

* **«Официант»** – сотрудник ресторана, принимающий заказ у клиента, консультирующий по меню, передающий заказ на кухню, осуществляющий подачу блюда и расчёт.
* **«Администратор»** – сотрудник, управляющий работой зала и кухни, отвечающий за корректность заказов, ведение журналов, сверку остатков и контроль выполнения.
* **«Повар»** – сотрудник кухни, осуществляющий приготовление блюда по технической карте на основании принятого заказа и при наличии необходимых ингредиентов.

В рамках данной функции осуществляется взаимодействие между клиентом и персоналом ресторана: от приёма запроса до передачи задания на кухню и выдачи квитанции клиенту.



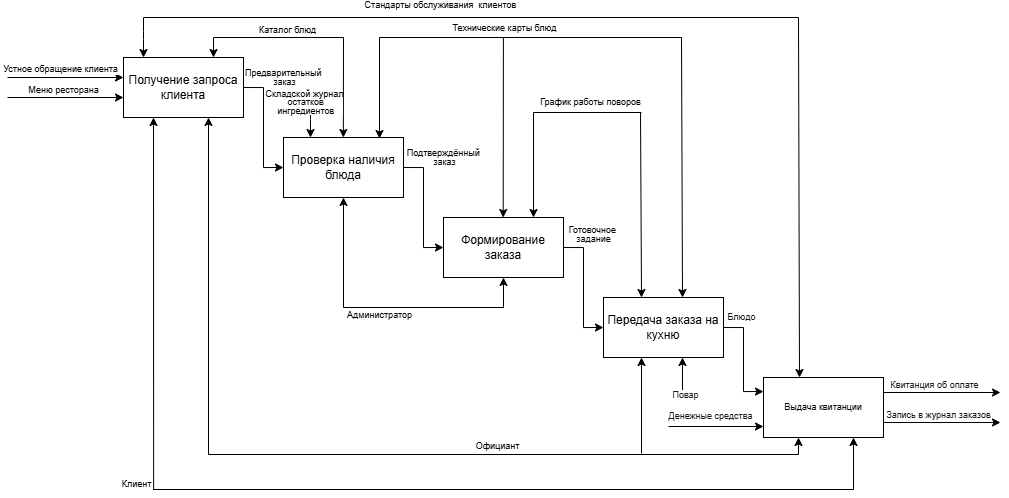
*Рисунок 2 – Контекстная диаграмма БП обработки заказа на блюдо*

#### Декомпозиция контекстной диаграммы первого уровня «Обработка заказа на блюдо»:

В результате декомпозиции первого уровня процесс разделяется на пять взаимосвязанных подфункций:

* A1.1 Получение запроса клиента;
* A1.2 Проверка наличия блюда;
* A1.3 Формирование заказа;
* A1.4 Передача заказа на кухню;
* A1.5 Выдача квитанции.

Декомпозиция первого уровня «Обработка заказа на блюдо»:



*Рисунок 3 – Декомпозиция бизнес-процесса «Обработка заказа на блюдо» первого уровня*

**Входы** — информация, подвергаемая обработке:

* **«Устное обращение клиента»**
* **«Меню ресторана» (бумажный каталог блюд)**
* **«Денежные средства»**

**Выходы** — данные, полученные в результате выполнения работы:

* **«Блюдо»**
* **«Квитанция об оплате»**
* **«Запись в журнал заказов»**

**Промежуточные объекты** — данные, которые служат входами для следующих функциональных блоков:

* **«Предварительный заказ»** — зафиксированная официантом информация о желаемых блюдах клиента, включая возможные уточнения (добавки, замены, особенности подачи). Используется администратором и поваром.
* **«Подтверждённый заказ» —** результат сверки с журналом остатков; содержит точную информацию о возможных к приготовлению блюдах. Используется для выдачи задания на кухню.
* **«Готовочное задание»** — внутренний рабочий документ, передаваемый на кухню; содержит наименование блюда, способ приготовления, время подачи и указания для повара.
* **«Отметка об оплате»** — признак завершённой оплаты по заказу; используется для закрытия заказа и учета выручки.
* **«Закрытый заказ»** — финальное состояние заказа, при котором блюдо приготовлено, подано, оплачено и зафиксировано в журнале.

**Управление** — условия, ограничения, указания:

* **«Каталог блюд»**
* **«Стандарты обслуживания клиентов»**
* **«Технические карты блюд»**
* **«График работы поваров»**
* **«Складской журнал остатков ингредиентов»**

**Механизмы или исполнители** — те, кто выполняет работу:

* **«Клиент»** — формулирует пожелания и производит оплату.
* **«Официант»** — принимает заказ, обслуживает клиента, передаёт блюдо, оформляет оплату.
* **«Администратор»** — контролирует остатки, организует передачу заказа, регистрирует данные.
* **«Повар»** — готовит блюдо в соответствии с заданием и тех. картой.

#### A1.1 Декомпозиция бизнес-процесса «Получение запроса клиента»



*Рисунок 4 – Декомпозиция бизнес-процесса «Получение запроса клиента» Ывторого уровня*

**Входы:**

* «Устное обращение клиента»
* «Меню ресторана»

**Выходы:**

* «Предварительный заказ»

**Промежуточные объекты:**

* «Заказ-карточка» — первичная бумажная форма, заполняемая официантом на основе пожеланий клиента.

**Управление:**

* «Каталог блюд»
* «Стандарты обслуживания клиентов»

**Механизмы:**

* «Клиент»
* «Официант»

#### A1.2 Декомпозиция бизнес-процесса «Проверка наличия блюда»



*Рисунок 5 – Декомпозиция бизнес-процесса «Проверка наличия блюда» второго уровня*

**Входы:**

* Предварительный заказ

**Выходы:**

* Подтверждённый заказ

**Промежуточные объекты:**

* Отчёт о наличии ингредиентов — результат сверки заказа с фактическими остатками продуктов на складе.

**Управление:**

* Складской журнал остатков ингредиентов
* Каталог блюд
* Технические карты блюд

**Механизмы:**

* Официант — при необходимости уточняет детали заказа у клиента на основании результатов проверки доступности блюд
* Администратор

#### A1.3 Декомпозиция бизнес-процесса «Формирование заказа»



*Рисунок 6 – Декомпозиция бизнес-процесса «Формирование заказа» второго уровня*

**Входы:**

Подтверждённый заказ

**Выходы:**

* Готовочное задание

**Промежуточные объекты:**

* Лист заданий на кухню — документ с параметрами приготовления и передачи блюда повару

**Управление:**

* Технические карты блюд
* График работы поваров

**Механизмы:**

* **Официант** — в случае необходимости взаимодействия с клиентом по результатам проверки
* Администратор

#### A1.4 Декомпозиция бизнес-процесса «Передача заказа на кухню»



*Рисунок 7 – Декомпозиция бизнес-процесса «Передача заказа на кухню» второго уровня*

**Входы:**

* Готовочное задание

**Выходы:**

* Блюдо

**Промежуточные объекты:**

* Маршрут заказа — устная или письменная передача задания на кухню, с указанием времени и особенностей приготовления

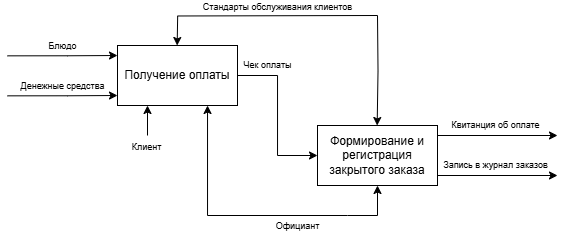
**Управление:**

* Технические карты блюд
* График работы поваров

**Механизмы:**

* Повар
* Официант

#### A1.5 Декомпозиция бизнес-процесса «Выдача квитанции»



*Рисунок 8 – Декомпозиция бизнес-процесса «Выдача квитанции» второго уровня*

**Входы:**

* Блюдо
* Денежные средства

**Выходы:**

* Квитанция об оплате
* Запись в журнал заказов

**Промежуточные объекты:**

* Чек оплаты — форма, подтверждающая факт получения оплаты
* Закрытый заказ — заказ, завершённый и зафиксированный в учётной документации ресторана

**Управление:**

* Стандарты обслуживания клиентов

**Механизмы:**

* Клиент
* Официант

### 1.3. Глоссарий предметной области

Таблица 1 – Глоссарий предметной области

|  |  |
| --- | --- |
| Название термина | Описание термина |
| Клиент | Посетитель ресторана, желающий заказать блюдо и получить обслуживание в зале. |
| Официант | Сотрудник ресторана, принимающий заказ от клиента, передающий его на кухню, подающий блюда и оформляющий счёт. |
| Администратор | Работник ресторана, отвечающий за бронирование столиков, ведение документации, контроль остатков, распределение заказов и взаимодействие с персоналом. |
| Повар | Работник кухни, отвечающий за приготовление блюд согласно поступившим заказам и техническим картам. |
| Журнал заказов | Бумажный журнал, в котором фиксируются все заказы клиентов: дата, состав, особенности, обслуживающий официант и сумма. |
| Журнал бронирований | Реестр, в который администратор вносит данные о резервировании столиков (дата, время, количество гостей). |
| Складской журнал | Документ с информацией об остатках продуктов и сырья, необходимых для приготовления блюд, ведётся вручную. |
| Реестр поставок | Бумажные накладные от поставщиков, содержащие перечень поставленных продуктов, даты и объёмы поставки. |
| Журнал продаж | Бумажный документ с ежедневной сводкой по выручке, числу обслуженных клиентов, составу проданных блюд и выданным квитанциям. |
|  | Продолжение таблицы 1 |
| Реестр претензий | Сборник бумажных жалоб, отзывов и предложений клиентов, подаваемых в ресторан. |
| Меню | Бумажный каталог с наименованиями, описаниями и ценами блюд, используемый официантом и клиентом для формирования заказа. |
| Техническая карта блюда | Внутренний бумажный документ, содержащий перечень ингредиентов и технологию приготовления конкретного блюда, используется поваром. |
| Заказ | Внутренний бумажный документ - заявка, оформленная официантом по итогам общения с клиентом, включающая состав блюда, примечания и расчёт суммы. |
| Квитанция | Бумажный документ, подтверждающий факт оплаты заказа, содержит сумму, дату и номер заказа. |
| Готовочное задание | Внутренний бумажный документ, содержащий состав заказа, время подачи и инструкции; передаётся на кухню повару. |
| Продукт | Запись информации на бумажном носителе - любой ингредиент, используемый на кухне для приготовления блюд, подлежит учёту в складском журнале. |
| Блюдо | Запись информации на бумажном носителе - единица продукции кухни, заказываемая клиентом; включает название, состав, цену и способ приготовления. |
| Ингредиент | Запись информации во внутреннем документе (технической карте) – продукт, входящий в состав блюда. |

### 1.4 Документ “Видение”

#### Совладельцы системы

Совладельцами информационной системы являются:

* **Администратор ресторана** – сотрудник, ответственный за приём клиентов, бронирование столиков, оформление заказов, ведение бумажных журналов (заказов, продаж, поставок и претензий), а также взаимодействие с поварами и официантами. После внедрения информационной системы будет являться основным пользователем, осуществляющим запись и обработку операций в электронной форме.
* **Официант** – сотрудник, принимающий устные заказы от клиентов, вручную заполняющий бланки, передающий заказы на кухню, оформляющий квитанции и расчёты. После автоматизации будет использовать систему для электронной регистрации заказов и расчётов.
* **Менеджер ресторана** – пользователь, заинтересованный в агрегированных данных по продажам, остаткам и эффективности работы. Будет использовать аналитические отчёты для контроля филиалов и принятия управленческих решений.

**Границы системы**

Разрабатываемая система охватывает процессы как на уровне одного ресторана, так и на уровне всей сети. Её границы определяются следующими рабочими местами:

* **Рабочее место администратора** – бронирование столиков, регистрация заказов, контроль остатков, приём и учёт поставок;
* **Рабочее место официанта** – оформление заказа, его передача на кухню, расчёт с клиентом;
* **Рабочее место управляющего** – просмотр отчётов, анализ продаж и остатков, управление данными филиалов.

Таблица 2 – Проблемы предметной области

|  |  |
| --- | --- |
| Элемент | Описание |
| Проблема | Отсутствие автоматизации ключевых процессов (приём заказов, бронирование, учёт остатков, поставки, сбор отзывов). Работа осуществляется вручную с использованием бумажных форм и журналов: официанты вручную фиксируют заказы, администраторы ведут учёт в журналах, отсутствует централизованный контроль над данными. |
| Воздействует на | Клиентов (долгое ожидание, путаница в заказах), официантов и администраторов (ручная работа, ошибки, перегрузка), поваров (отсутствие точной информации), менеджмент (невозможность анализа данных и планирования). |
| Результатом чего является | Ошибки в обслуживании (несовпадения в заказах, недоступность блюд), потери продуктов, задержки в обслуживании, недовольство клиентов, снижение качества сервиса, отсутствие прозрачной статистики для управления и развития сети. |
| Выигрыш от внедрения ИС | * Централизованный учёт заказов, остатков и поставок * Повышение точности данных и уменьшение ошибок * Сбор и хранение отзывов для оценки качества * Доступ к статистике и аналитике для руководства * Повышение эффективности бизнес-процессов и конкурентоспособности сети |

**Возможности системы**

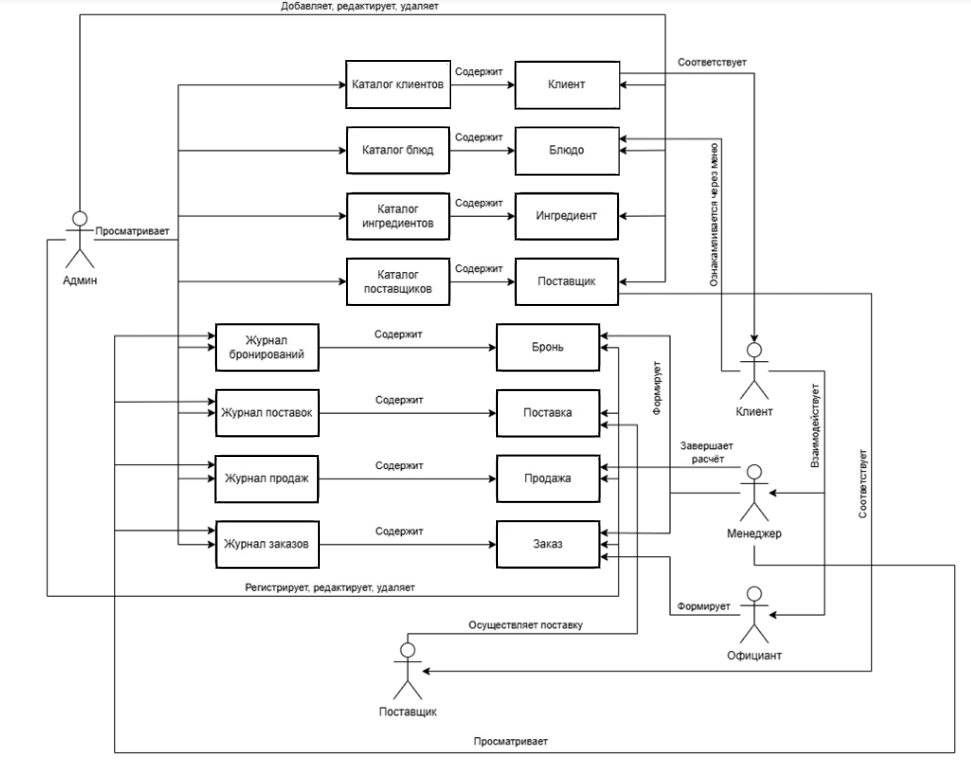
Информационная система будет включать в себя следующие ключевые возможности:

* Ввод, хранение и редактирование информации о заказах клиентов (набор блюд, примечания, расчёт суммы, статус оплаты);
* Ведение электронной базы бронирований с возможностью контроля по дате, времени и числу гостей;
* Учёт складских остатков с возможностью автоматического обновления на основе заказов и поставок;
* Электронный реестр поставок с фиксацией даты, объёма и состава каждой поставки;
* Поддержка каталога меню с возможностью добавления, изменения и удаления позиций блюд;
* Хранение данных о поставщиках с функциями редактирования и поиска;
* Сбор отзывов клиентов в цифровой форме с возможностью анализа качества обслуживания;
* Формирование рейтингов блюд и выявление наиболее востребованных позиций;
* Анализ загруженности кухни и определение пиковых временных интервалов;
* Генерация аналитических отчётов по продажам, остаткам, поставкам и эффективности филиалов;
* Централизованное управление сетью ресторанов, включая как собственные, так и франчайзинговые точки;
* Поиск по заказам, блюдам, клиентам, поставкам и другим сущностям по заданным параметрам.

## Я ща2. Анализ

### 2.1 Концептуальная модель предметной области

Для формализации предметной области построена концептуальная UML-диаграмма классов [1], отражающая основные сущности системы, их свойства и взаимосвязи. Модель разработана на основе анализа текущих процессов в ресторане, выполняемых вручную с использованием бумажных документов.



*Рисунок 9 – Концептуальная модель предметной области информационной системы ресторана*

### 2.2. Функциональные требования

Таблица 3 – Функциональные требования

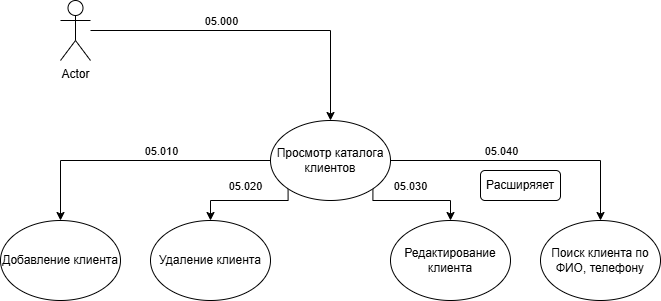
|  |  |
| --- | --- |
| Номер | Функциональное требование |
| FR010 | Система должна позволять просматривать список бронирований. |
| FR020 | Система должна позволять добавлять новые бронирования. |
| FR030 | Система должна позволять редактировать и удалять бронирования. |
| FR040 | Система должна отображать только доступные для брони столики. |
| FR050 | Система должна позволять клиенту или официанту формировать корзину до оформления заказа. |
| FR060 | Система должна проверять наличие ингредиентов перед оформлением заказа. |
| FR070 | Система должна позволять сохранять историю заказов клиента. |
| FR080 | Система должна отображать рейтинг блюд на основе частоты заказов. |
| FR090 | Система должна формировать рейтинг поваров на основе отзывов и статистики. |
| FR100 | Система должна отображать текущую загруженность кухни. |
| FR110 | Система должна позволять просматривать информацию о заказе. |
|  | Продолжение таблицы 3 |
| FR120 | Система должна позволять вводить информацию о новом заказе. |
| FR130 | Система должна позволять удалять информацию о заказе. |
| FR140 | Система должна позволять обновлять информацию о заказе. |
| FR150 | Система должна отображать статус заказа (новый, в работе, выполнен, оплачен). |
| FR160 | Система должна автоматически рассчитывать итоговую сумму заказа. |
| FR170 | Система должна позволять искать заказы по клиенту, дате, ресторану и статусу. |
| FR180 | Система должна позволять управлять информацией о ресторанах (добавление, удаление, редактирование). |
| FR190 | Система должна позволять искать рестораны по названию, адресу и телефону. |
|  |  |
|  |  |
|  | Продолжение таблицы 3 |
| FR200 | Система должна хранить полную информацию о ресторане:   * Название; * Адрес; * Телефон; * Ответственное лицо. |
| FR210 | * Система должна позволять управлять информацией о блюдах (добавление, удаление, редактирование). |
| FR220 | Система должна отображать состав блюда и его категорию. |
| FR230 | Система должна позволять искать блюда по названию, категории, цене. |
| FR240 | Система должна позволять управлять списком поваров (добавление, удаление, редактирование). |
| FR250 | Система должна отображать информацию о поваре:   * ФИО; * Табельный номер; * Опыт; * Специализация. |
| FR260 | * Система должна позволять искать поваров по опыту, имени, рейтингу. |
| FR270 | Система должна позволять регистрировать клиентов и редактировать их данные. |
| FR280 | Система должна отображать информацию о клиенте:   * ФИО * Телефон |
|  | Продолжение таблицы 3 |
|  | * История заказов. |
| FR290 | Система должна позволять искать клиентов по имени и телефону. |
| FR300 | Система должна регистрировать поставки продуктов с данными о дате, поставщике и составе. |
| FR310 | Система должна обновлять складские остатки при приёме поставок. |
| FR320 | Система должна позволять управлять данными поставщиков. |
| FR330 | Система должна отображать журнал поставок по всем филиалам. |
| FR340 | Система должна отображать журнал ингредиентов в ресторанах. |
| FR350 | Система должна позволять анализировать данные о заказах, продажах и остатках. |
| FR360 | Система должна формировать отчёты по популярности блюд, загрузке кухни и выручке. |
| FR370 | Система должна отображать и регистрировать обращения клиентов. |
| FR380 | Система должна позволять руководству просматривать сводную аналитику по всей сети ресторанов. |

### 2.3. Диаграммы вариантов использования

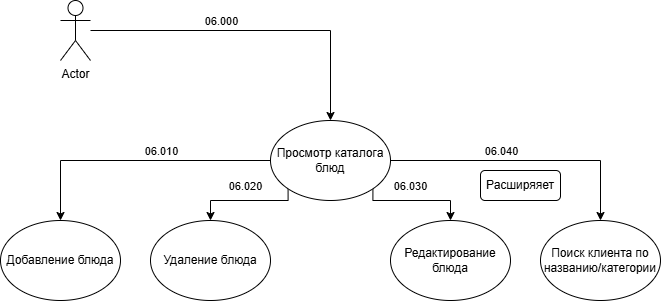
Примеры диаграмм вариантов использования информационной системы ресторана:



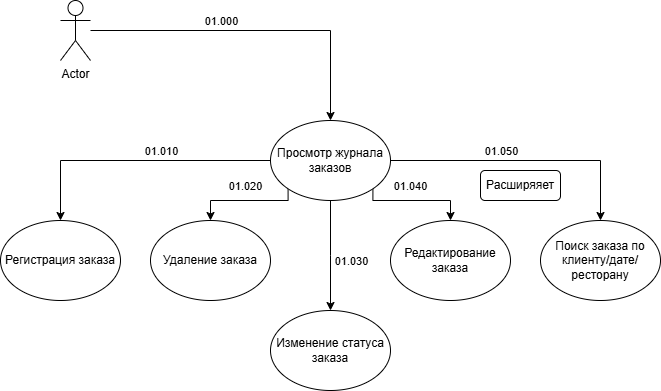
*Рисунок 10 – Неполная диаграмма вариантов использования системы для администратора ресторана*



*Рисунок 11 – Диаграмма варианта использования (прецедента) просмотр каталога клиентов*



*Рисунок 12 – Диаграмма варианта использования (прецедента) просмотр каталога блюд*



*Рисунок 13 – Диаграмма варианта использования (прецедента) просмотр журнала заказов*

### 2.4 Краткое и подробное описание вариантов использования

#### Вариант использования: Просмотр информации о клиентах

**Описание:**  
 Данный прецедент описывает процесс просмотра администратором информации о клиентах. Система предоставляет таблицу с основными сведениями о клиентах, включая ФИО, контактный телефон и историю заказов. Прецедент также предполагает действия по управлению клиентской информацией: добавление, редактирование, удаление записей и поиск по ключевым параметрам.

**Требования:** FR270, FR280, FR290, FR300, FR310

**Основной ход событий:**

Таблица 4 – Основной ход событий просмотра информации о клиентах

|  |  |
| --- | --- |
| Действия актёра | Отклик системы |
| 10. Актёр переходит в раздел «Клиенты» |  |
|  | 20. Система отображает таблицу с записями клиентов |
| 30. Актёр просматривает ФИО, контактные данные и историю заказов |  |
|  | Продолжение таблицы 4 |
|  | 40. Система предоставляет опции сортировки и фильтрации |
| 50. Актёр может выбрать клиента для редактирования или удаления |  |
|  | 60. Система открывает соответствующий интерфейс и выполняет операцию |

**Альтернативный ход событий:**

* Если в базе данных отсутствуют клиенты, система выводит сообщение:  
  «Список клиентов пуст».
* В случае сбоя соединения с базой данных система отображает сообщение об ошибке:  
  «Ошибка подключения. Повторите попытку позже».

**Точки расширения:**

Таблица 5 – Точки расширения просмотра информации о клиентах

|  |  |  |
| --- | --- | --- |
| Название | Описание | Вариант использования |
| Добавление клиента | Переход к форме добавления новой записи | 05.010 |
|  | Продолжение таблицы 5 |  |
| Удаление клиента | Удаление выделенной записи клиента | 05.020 |
| Редактирование клиента | Изменение данных выбранного клиента | 05.030 |
| Поиск клиента по ФИО или телефону | Поиск среди записей по заданным параметрам (расширяет основной сценарий) | 05.040 |

#### Вариант использования: Просмотр журнала заказов

**Описание:**  
 Данный прецедент описывает взаимодействие администратора с системой для просмотра журнала заказов. Система предоставляет таблицу заказов с возможностью просмотра состава, даты оформления, клиента, ресторана, суммы и статуса. В рамках данного варианта использования предусмотрены функции регистрации, удаления, редактирования и изменения статуса заказов. Также доступен поиск по ключевым параметрам.

**Требования:** FR010 – FR040, FR170, FR210, FR320

**Основной ход событий:**

Таблица 6 – Основной ход событий просмотра журнала заказов

|  |  |
| --- | --- |
| Действия актёра | Отклик системы |
| 10. Актёр выбирает пункт «Журнал заказов» |  |
|  | 20. Система отображает таблицу заказов |
| 30. Актёр просматривает поля: № заказа, клиент, дата, блюда, статус, сумма |  |
|  | 40. Доступны функции фильтрации, сортировки и перехода к редактированию заказа |
| 50. Актёр может выбрать запись и выполнить одно из действий: редактирование, удаление, изменение статуса |  |
|  | 60. Система открывает соответствующий интерфейс или выполняет действие |

#### Альтернативный ход событий:

* Если в системе отсутствуют заказы, отображается сообщение:  
  «Журнал заказов пуст»
* В случае технической ошибки при загрузке данных выводится сообщение:  
  «Ошибка загрузки журнала. Повторите попытку позже»

**Точки расширения:**

Таблица 7 – Точки расширения просмотра информации о клиентах

|  |  |  |
| --- | --- | --- |
| Название | Описание | Вариант использования |
| Регистрация заказа | Открытие формы создания нового заказа | 01.010 |
| Удаление заказа | Удаление выбранной записи | 01.020 |
| Изменение статуса заказа | Установка нового статуса (новый, на кухне, выполнен, оплачен и др.) | 01.030 |
| Редактирование заказа | Переход к редактированию состава и параметров заказа | 01.040 |
| Поиск заказа по клиенту/дате/ресторана | Расширяет базовый сценарий, позволяет фильтровать записи по параметрам | 01.050 (<<расширяет>>) |

#### Вариант использования: Просмотр каталога блюд

**Описание:**  
Данный прецедент описывает взаимодействие администратора с системой для просмотра текущего перечня блюд, представленных в меню ресторана. В процессе просмотра отображаются наименование, категория, состав и цена каждого блюда. Также предоставляется возможность управления записями (добавление, удаление, редактирование) и поиска по ключевым параметрам.

**Требования:** FR090, FR100, FR110, FR120, FR190, FR220

**Основной ход событий:**

Таблица 8 – Основной ход событий просмотра каталога блюд

|  |  |
| --- | --- |
| Действия актёра | Отклик системы |
| 10. Актёр выбирает пункт «Каталог блюд» |  |
|  | 20. Система отображает таблицу с данными по всем блюдам |
| 30. Актёр просматривает поля: название, категория, цена, состав |  |
|  | 40. Система предоставляет опции сортировки и фильтрации по параметрам |
| 50. Актёр может инициировать действия по добавлению, редактированию или удалению блюда |  |
|  | 60. Система открывает соответствующие формы |

#### Альтернативный ход событий:

Если в базе отсутствуют записи о блюдах, система отображает сообщение: *«Каталог блюд пуст. Добавьте хотя бы одно блюдо.»*

При ошибке доступа к базе данных система выводит сообщение: *«Ошибка загрузки каталога. Повторите попытку позже.»*

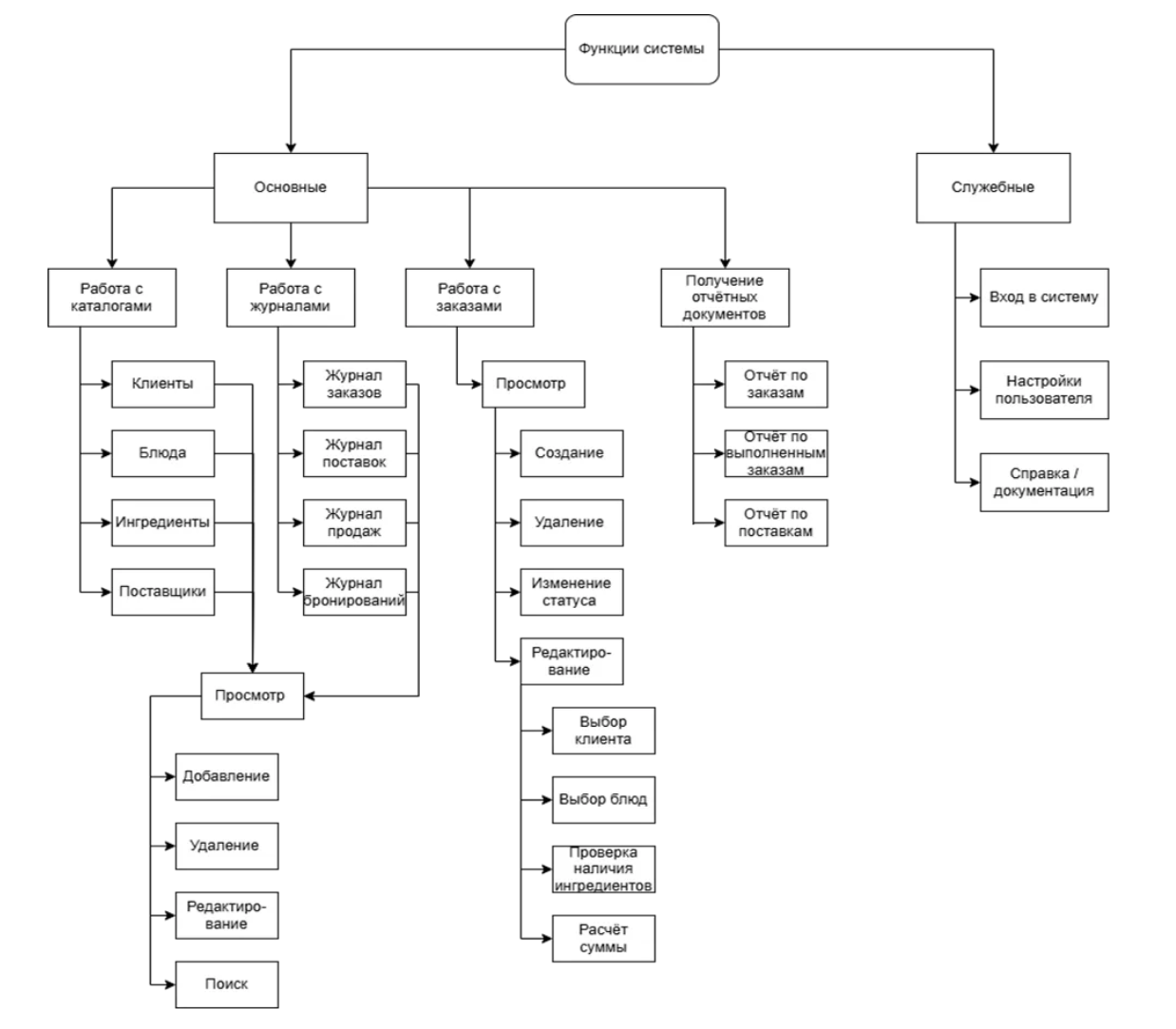
Точки расширения:

Таблица 9 – Точки расширения просмотра каталога блюд

|  |  |  |
| --- | --- | --- |
| Название | Описание | Вариант использования |
| Добавление блюда | Открытие формы создания новой записи | 06.010 |
| Удаление блюда | Удаление выбранной записи | 06.020 |
| Редактирование блюда | Переход к редактированию информации о блюде | 06.030 |

### 2.5. Проектирование пользовательского интерфейса

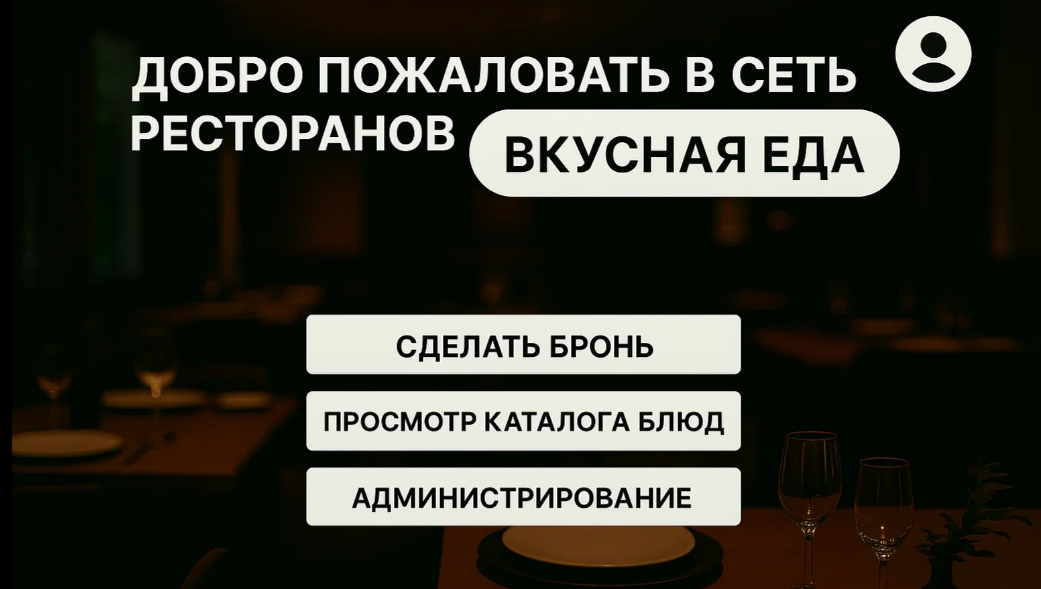
#### Дерево функций

 *Рисунок 14 – Дерево функций информационной системы*

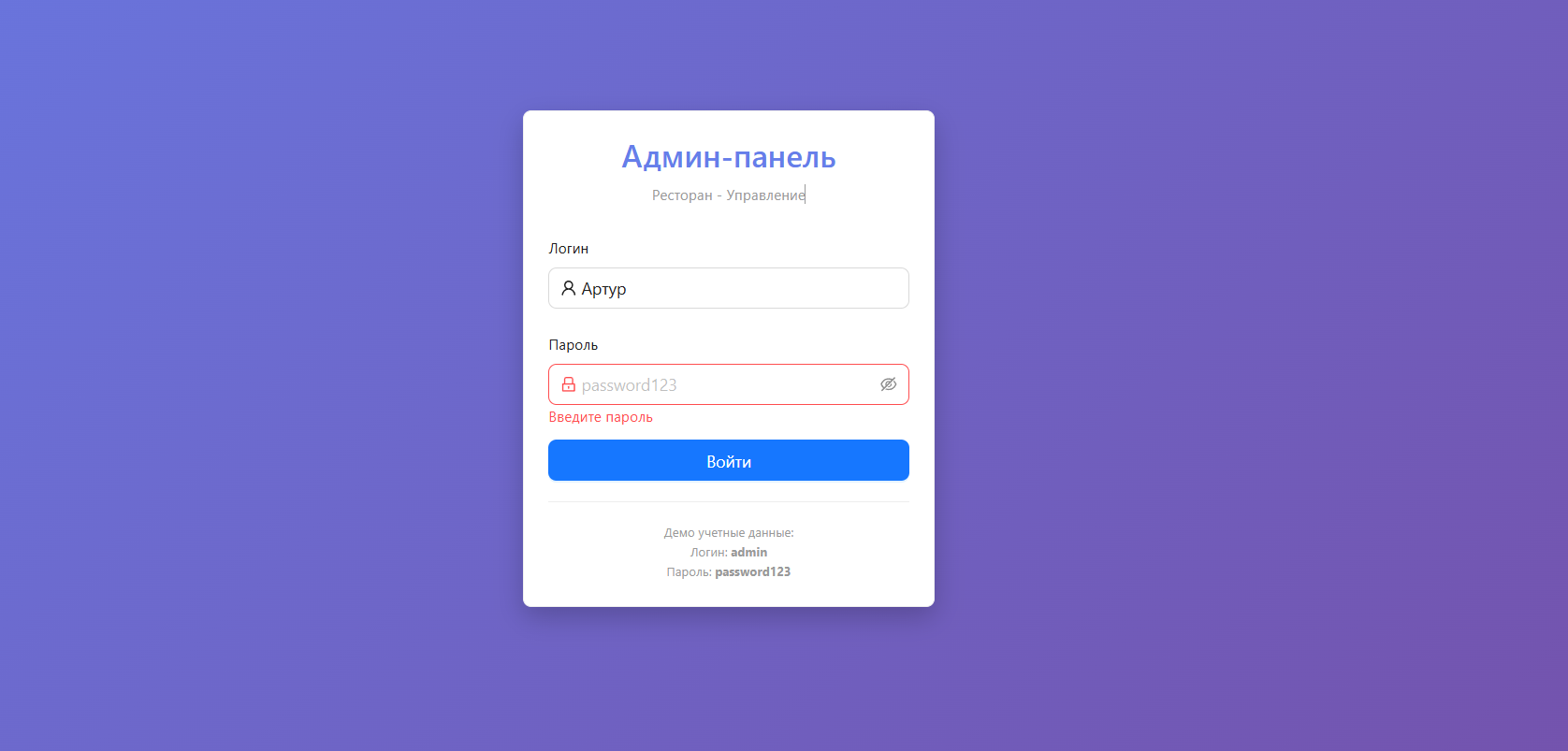
#### Пример сценария диалога с информационной системой

*Рисунок 15 – Пример сценария диалога с системой*

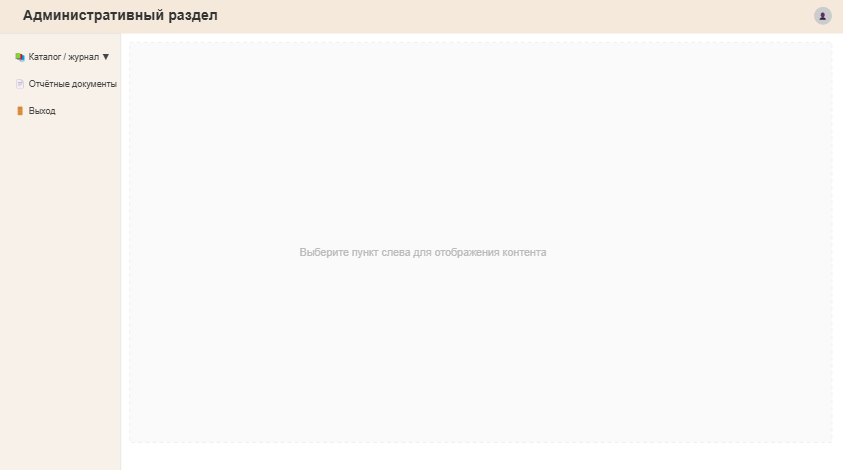
#### Макеты интерфейсов:



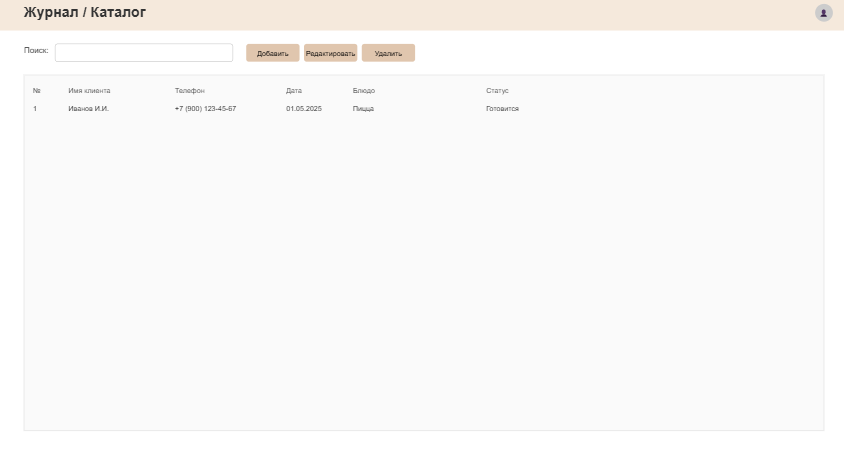
*Рисунок 16 – Макет главной страницы*



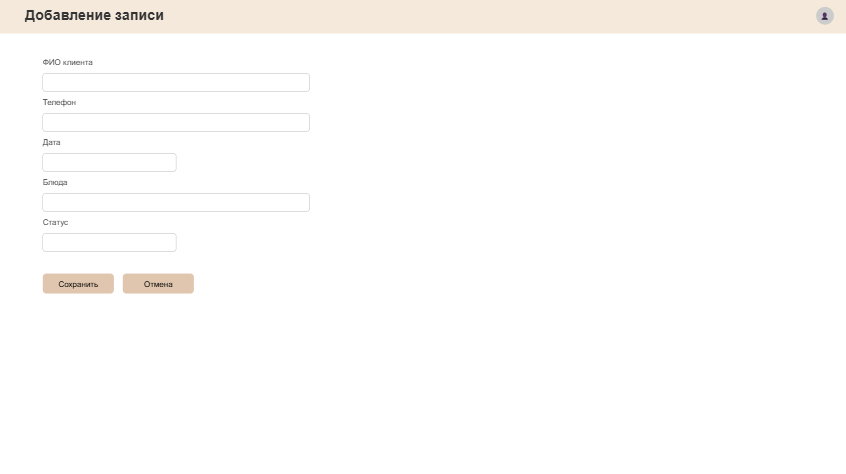
*Рисунок 17 – Макет страницы авторизации*



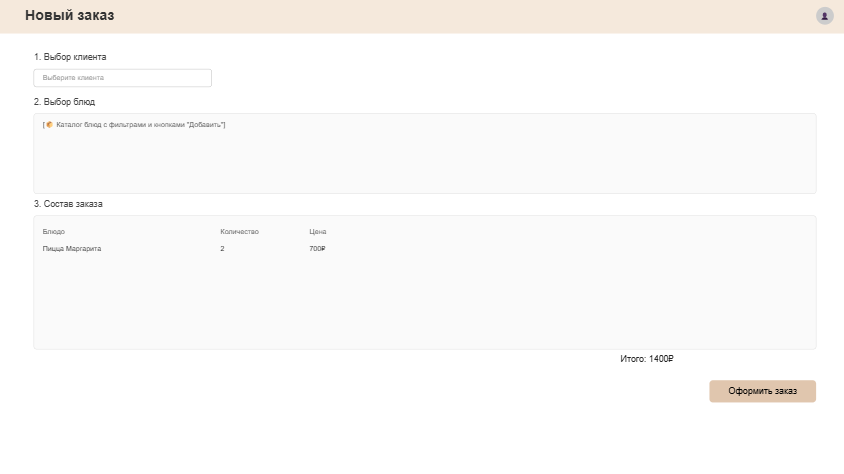
*Рисунок 18 – Макет страницы администрирования*



*Рисунок 19 – Макет страницы работы с каталогами и журналами*



*Рисунок 20 – Макет страницы создания брони*

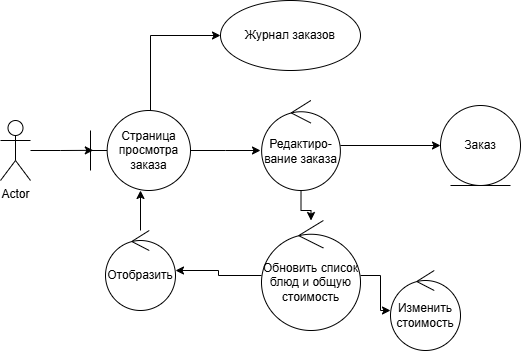


*Рисунок 21 – Макет страницы создания заказа*

## 3. Проектирование

### 3.1. Диаграммы анализа

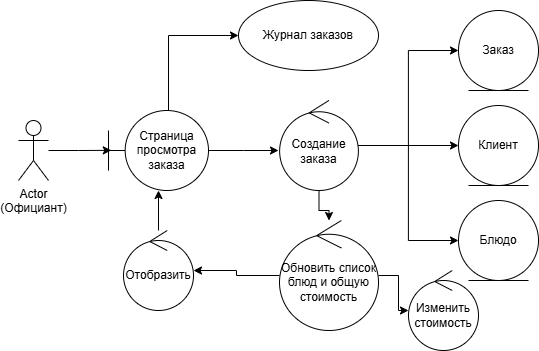
Диаграммы анализа (Analysis Diagrams) используются для уточнения требований и детального понимания взаимодействия объектов внутри системы. Они служат промежуточным звеном между описанием вариантов использования и финальной архитектурой, позволяя выявить ключевые классы сущностей, граничные классы и управляющие классы.



*Рисунок 22 – Диаграмма анализа варианта использования «Редактирование заказа 01.040»*



*Рисунок 23 – Диаграмма анализа варианта использования «Создание бронирования 04.010»*

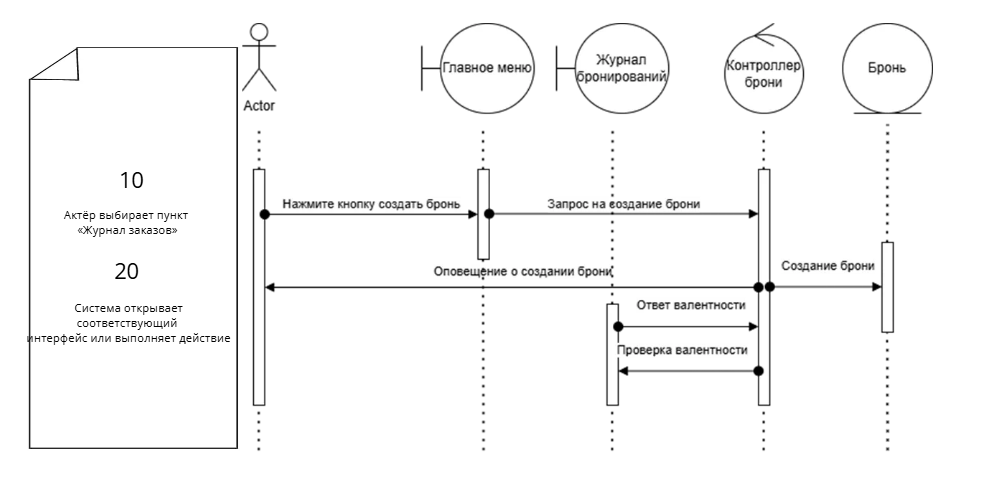


*Рисунок 24 – Диаграмма анализа варианта использования «Формирование заказа 01.010»*



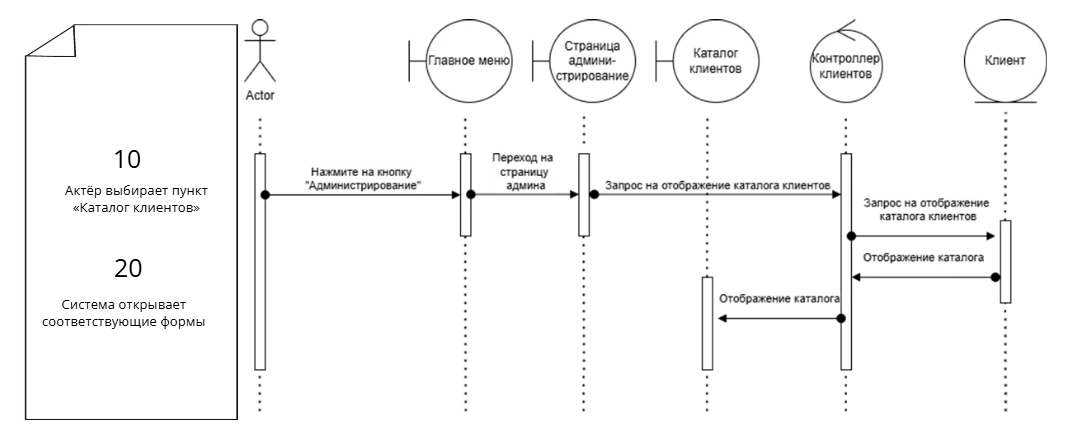
*Рисунок 25 –Диаграмма анализа варианта использования «Просмотр информации о клиентах 05.000»*

### 3.2. Диаграммы последовательности



*Рисунок 26 – Диаграмма последовательности для прецедента «Создание брони 04.010»*

На рисунке 26 в качестве журнала могут быть: журнал бронирований, журнал заказов, складской журнал, журнал продаж.

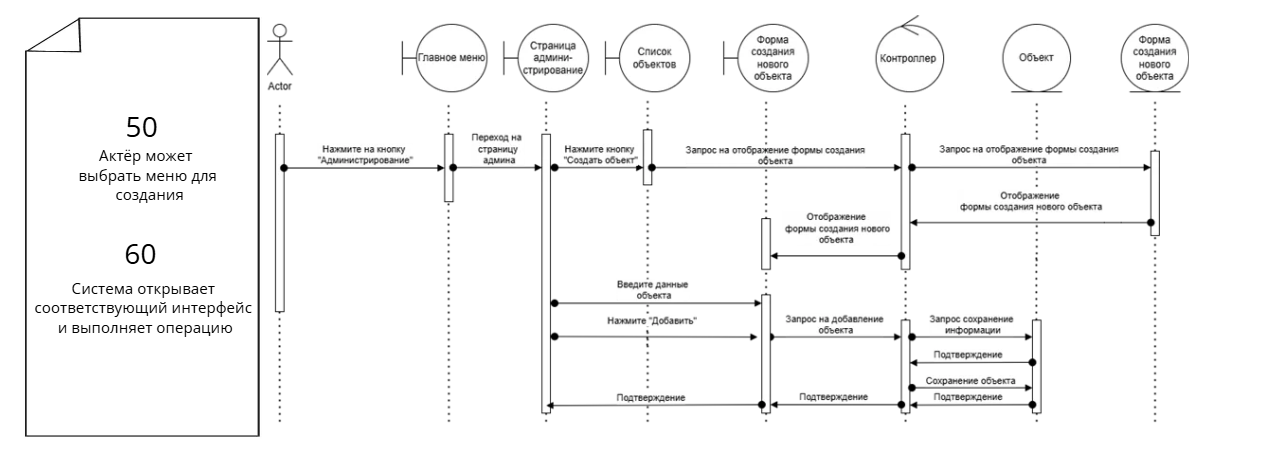


*Рисунок 27 – Диаграмма последовательности для прецедента «Просмотр каталога клиентов 05.000»*

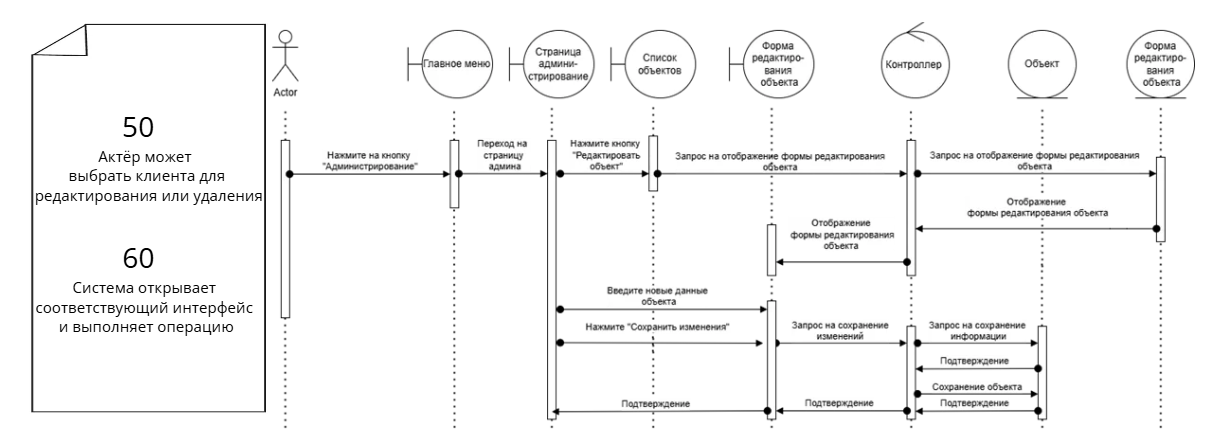
На рисунке 27 в качестве каталога могут быть: каталог клиентов, каталог блюд(меню), каталог жалоб (реестр претензий), каталог поставок (реестр поставок).

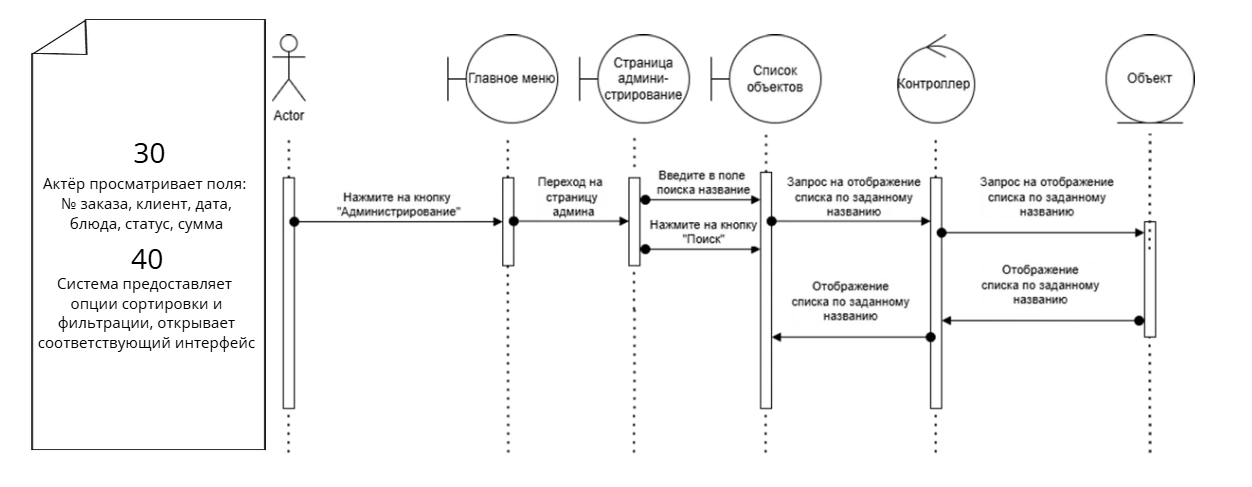


*Рисунок 28 – Диаграмма последовательности для прецедента «Удаление объекта из каталога/журнала»*



*Рисунок 29 – Диаграмма последовательности для прецедента «Поиск объекта»*

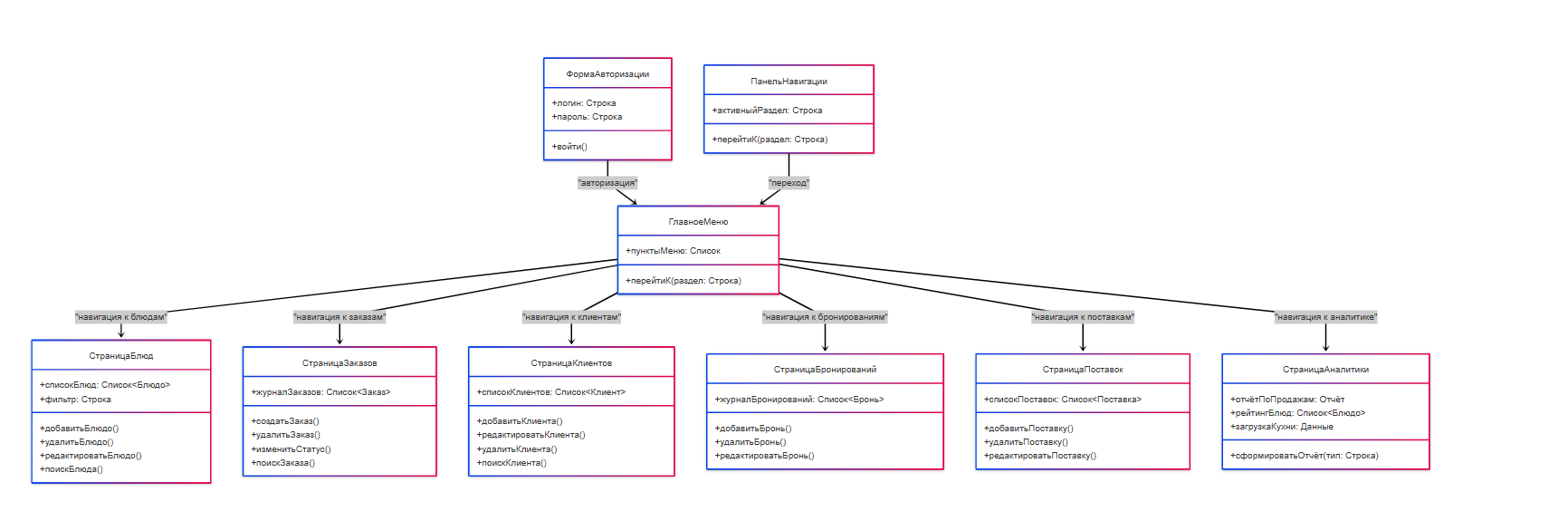
 *Рисунок 30 – Диаграмма последовательности для прецедента “Создание объекта”*



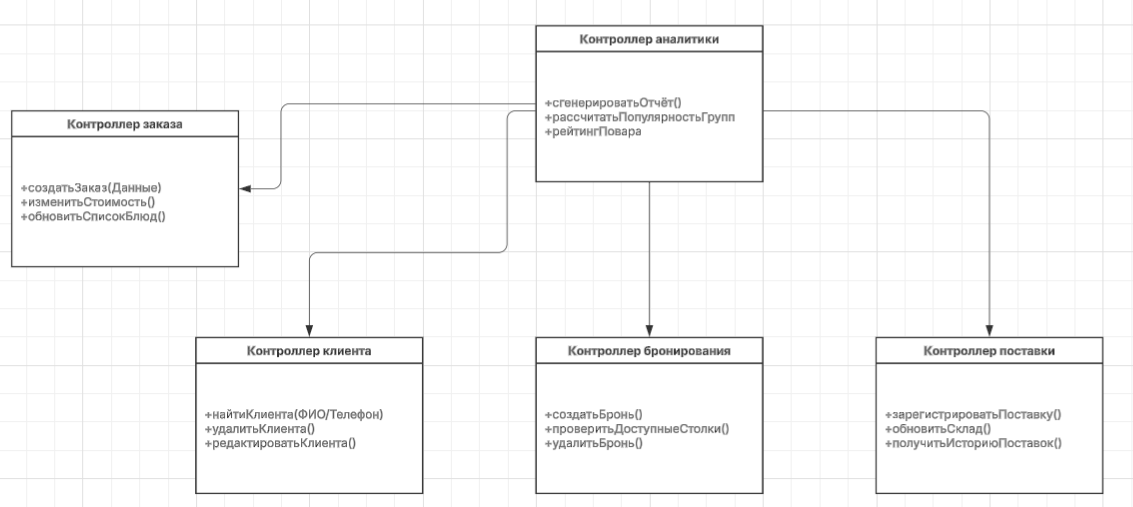
*Рисунок 31 – Диаграмма последовательности для прецедента «Сохранение объекта»*

### 3.3. Диаграммы классов этапа проектирования

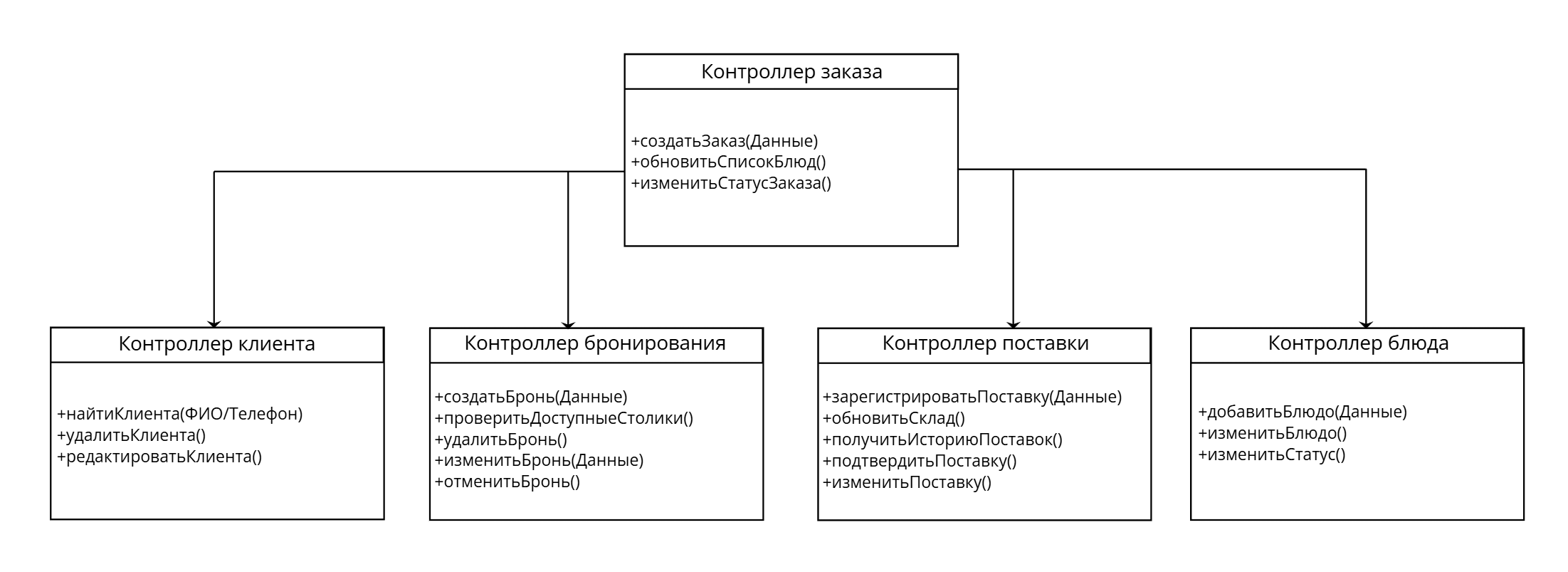
Разработанные диаграммы классов отражают структуру программного обеспечения на уровнях представления, прикладного и хранения данных. Для построения UML-диаграмм использовался инструмент PlantUML, позволяющий описывать структуру классов в виде текстового кода [9].



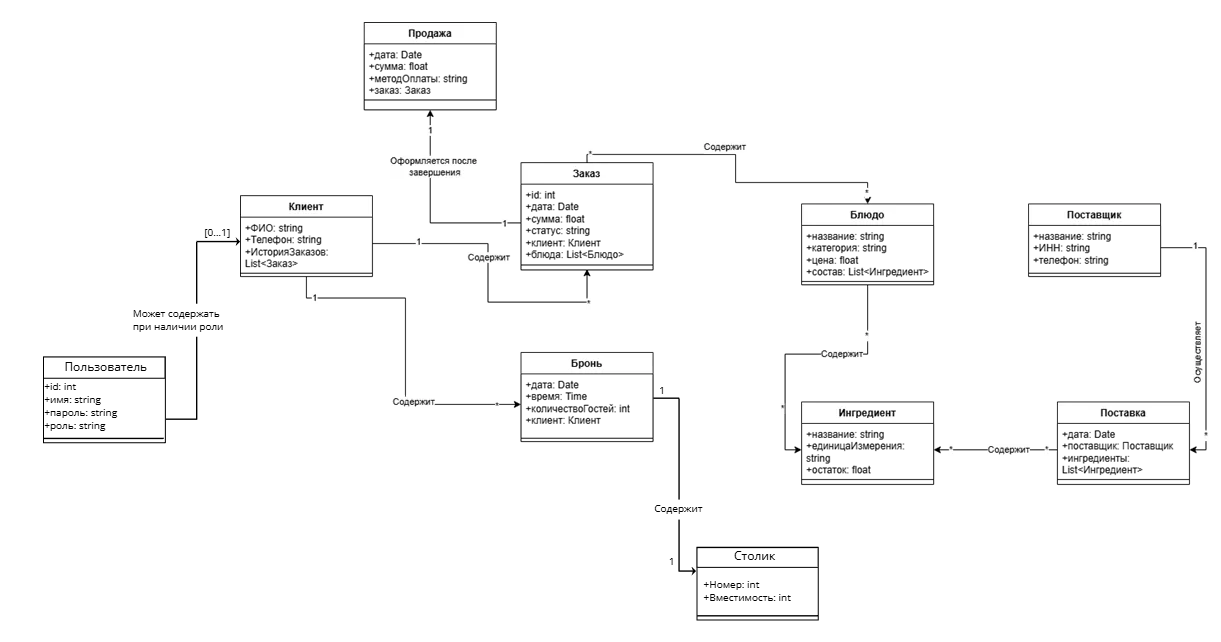
*Рисунок 32 – Диаграмма классов уровня представления*



*Рисунок 33 – Диаграмма классов прикладного уровня, первая часть отношения контроллеров*



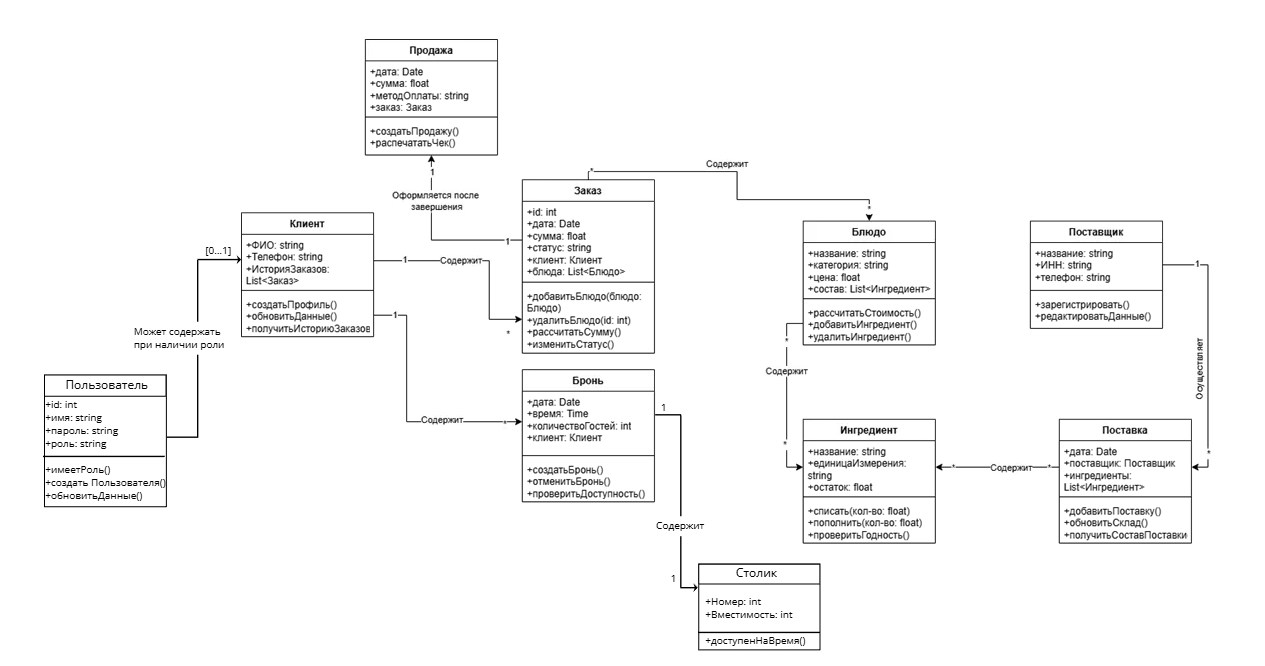
*Рисунок 34 – Диаграмма классов прикладного уровня, вторая часть отношения контроллеров*



*Рисунок 35 – Диаграмма классов уровня хранения данных (базы данных)*

### 3.4. Схема (структура) базы данных

Схема базы данных, спроектированная при помощи графического редактора [10]:



*Рисунок 36 – Диаграмма классов базы данных*

#### Переход от концептуальной к логической модели

На основании анализа предметной области и ранее разработанной концептуальной UML-диаграммы классов сформирована логическая структура базы данных. Концептуальная модель отражала сущности и связи между ними: Клиент, Заказ, Блюдо, Ингредиент, Поставка, Поставщик, Продажа, Бронь и др. Однако для реализации в реляционной СУБД потребовалось провести нормализацию, устранить дублирование и реализовать связи через внешние ключи.

#### Нормализация таблиц

В процессе проектирования логической модели применены нормальные формы, описанные в теории реляционных баз данных [3, 4]:

* **Первая нормальная форма (1НФ):**
  + Все атрибуты атомарны, без составных или повторяющихся групп.
  + Исключены вложенные структуры (например, список блюд в заказе), вынесены в отдельные таблицы.
* **Вторая нормальная форма (2НФ):**
  + Все неключевые атрибуты зависят от всего составного ключа.
  + Для этого были созданы вспомогательные сущности, такие как СоставЗаказа, СоставБлюда, СоставПоставки.
* **Третья нормальная форма (3НФ):**
  + Устранены транзитивные зависимости.
  + Например, информация о клиенте и поставщике вынесена в отдельные таблицы.
* **Четвёртая нормальная форма (4НФ):**
  + Устранены многозначные зависимости.
  + Связи многие-ко-многим реализованы через промежуточные таблицы, содержащие количественные характеристики (норма, объём, количество и т.д.).

#### Причины создания дополнительных таблиц

Таблица 10 – Причины создания дополнительных таблиц

|  |  |
| --- | --- |
| Таблица | Назначение |
| СоставЗаказа | Устранение связи M:N между сущностями *Заказ* и *Блюдо* |
| СоставБлюда | Устранение связи M:N между *Блюдо* и *Ингредиент* |
| СоставПоставки | Устранение связи M:N между *Поставка* и *Ингредиент* |

Создание этих таблиц позволило обеспечить гибкость структуры и избежать избыточности хранения данных.

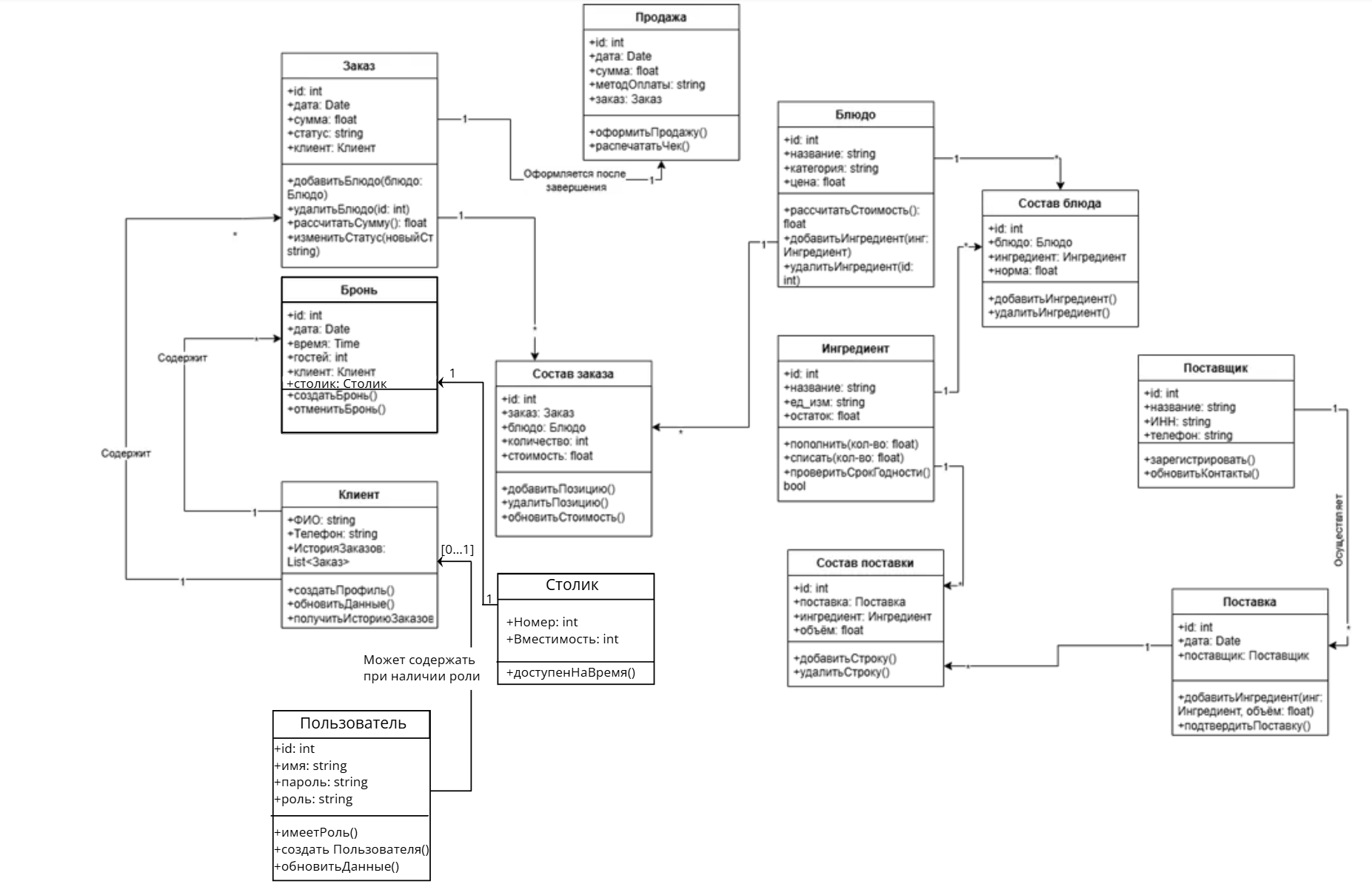
#### Итоговая структура реляционной модели

Разработанная логическая модель удовлетворяет требованиям нормализации до 4НФ, что обеспечивает:

* логическую целостность;
* отсутствие аномалий вставки, удаления и обновления;
* масштабируемость базы данных.

Диаграммы, представленные на рисунках 35 и 36, спроектированные с помощью инструментария [11], визуализируют переход от концептуальной к логической модели и демонстрируют корректную реализацию связей между сущностями.

Также были добавлены таблицы столиков и пользователей сайта для создания брони посетителем.



*Рисунок 37 – Физическая модель базы данных*

Разработанная физическая модель удовлетворяет требованиям нормализации до 4НФ.

## 4. Создание

Этап “Создание” является этапом реализации спроектированной информационной системы. В рамках данного этапа осуществляется выбор архитектурных решений, создание физической базы данных, написание программного кода системы, а также ее тестирование и отладка.

Особое внимание на данном этапе уделяется качеству кода и соблюдению принципов SOLID. Применение современных паттернов проектирования и ORM-технологий позволяет создать гибкую систему, готовую к будущим изменениям требований бизнеса без необходимости переписывания ядра приложения.

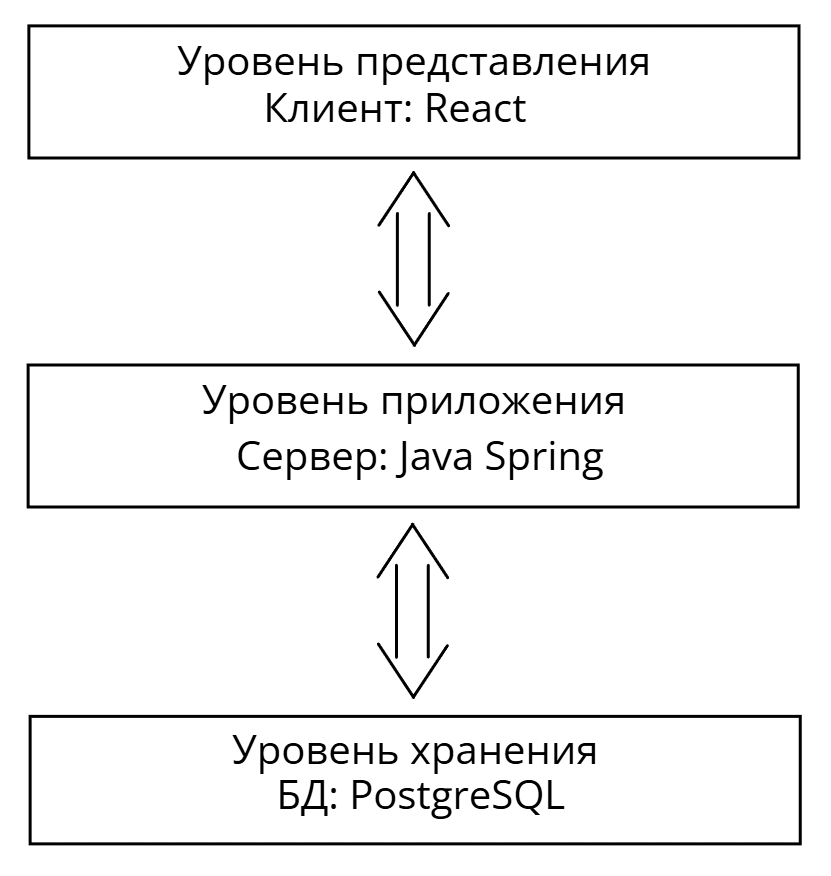
### 4.1. Выбор архитектуры программного обеспечения

Исходя из задания научно-исследовательской работы, для реализации информационной системы ресторана была выбрана **трёхуровневая архитектура**, которая предполагает разделение программного обеспечения на три независимых уровня для удобства и масштабирования разработки:

1. Уровень представления (Presentation Layer): обеспечивает интерфейс пользователя, визуализацию данных и обработку первичных действий пользователя;
2. Уровень приложения (Application Layer / Business Logic): реализует всю основную бизнес-логику системы, а также правила обработки данных и управление транзакциями;
3. Уровень хранения данных (Data Layer): обеспечивает надёжное хранение информации (данных) и доступ к базе данных.

Такое разделение повышает гибкость системы: изменения в интерфейсе не затрагивают логику хранения данных, а смена СУБД не требует переписывания пользовательского интерфейса, все части системы могут независимо изменяться.

На рисунке 38 показана схема реализации трехуровневой архитектуры в разработанной системе ресторана:

**

*Рисунок 38 – Реализация трехуровневой архитектуры системы*

В разработанной системе взаимодействие между уровнями организовано следующим образом:

* Уровень представления (веб-браузер) отправляет HTTP-запросы к уровню приложения;
* Уровень приложения (сервер) обрабатывает запрос, выполняет вычисления и при необходимости обращается к уровню хранения;
* Уровень хранения (СУБД) выполняет выборку или изменение данных и возвращает результат серверу.

### 4.2. Используемые инструменты и технологии веб-приложения

Для реализации системы были выбраны современные инструменты разработки, позволяющие эффективно воплотить трехуровневую архитектуру.

**Для уровня представления (Frontend):**

Выбрана библиотека React (с использованием языка TypeScript). Использование компонентного подхода React позволяет создавать динамичный интерфейс, где перерисовка страниц происходит без перезагрузки (Single Page Application), что повышает скорость работы персонала. Для управления состоянием приложения используется библиотека Zustand, а для взаимодействия с сервером – библиотека Axios.

**Для уровня приложения (Backend):**

Выбрана платформа Java с использованиемSpring Boot Framework.

* Spring Web MVC: обеспечивает обработку REST-запросов от клиента.
* Spring Security: отвечает за аутентификацию и авторизацию (используются JWT-токены).
* Spring Data JPA: предоставляет удобный интерфейс для работы с данными, абстрагируя SQL-запросы.

**Для уровня хранения данных:**

Выбрана реляционная СУБД PostgreSQL. Она обеспечивает соответствие требованиям ACID, поддерживает сложные связи между таблицами и высокую надежность хранения данных.

### 4.3. Создание базы данных

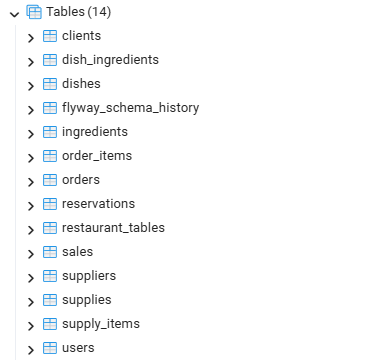
На основе спроектированной физической модели данных была реализована база данных restaurant\_db.

Процесс создания включал следующие шаги:

1. **Развертывание СУБД:** использовалась технология контейнеризации Docker для быстрого запуска экземпляра PostgreSQL;
2. **Создание таблиц:** написаны и выполнены SQL-скрипты (DDL), создающие структуру таблиц с учетом нормализации до 4НФ:

* Реализованы таблицы основных сущностей: users (пользователи), dishes (блюда), ingredients (ингредиенты), orders (заказы), sales (продажи), supplies (поставки), suppliers(поставщики), reservations (бронирования), restaurant\_tables (столики ресторана);
* Реализованы таблицы-связки для отношений «многие-ко-многим»: dish\_ingredients (состав блюд), order\_items (состав заказа), supply\_items (состав поставки).

1. **Ограничения целостности:** Добавлены внешние ключи (FOREIGN KEY) для обеспечения ссылочной целостности (нельзя удалить ингредиент, который используется в блюде).



*Рисунок 39 – Структура реализованной базы данных*

### 4.4. Создание программы системы

Разработка программного обеспечения велась в соответствии с принципами объектно-ориентированного программирования, выбранной архитектурой и требованиями к оформлению текста программ [7].

**Реализация уровня приложения (Серверная часть):**

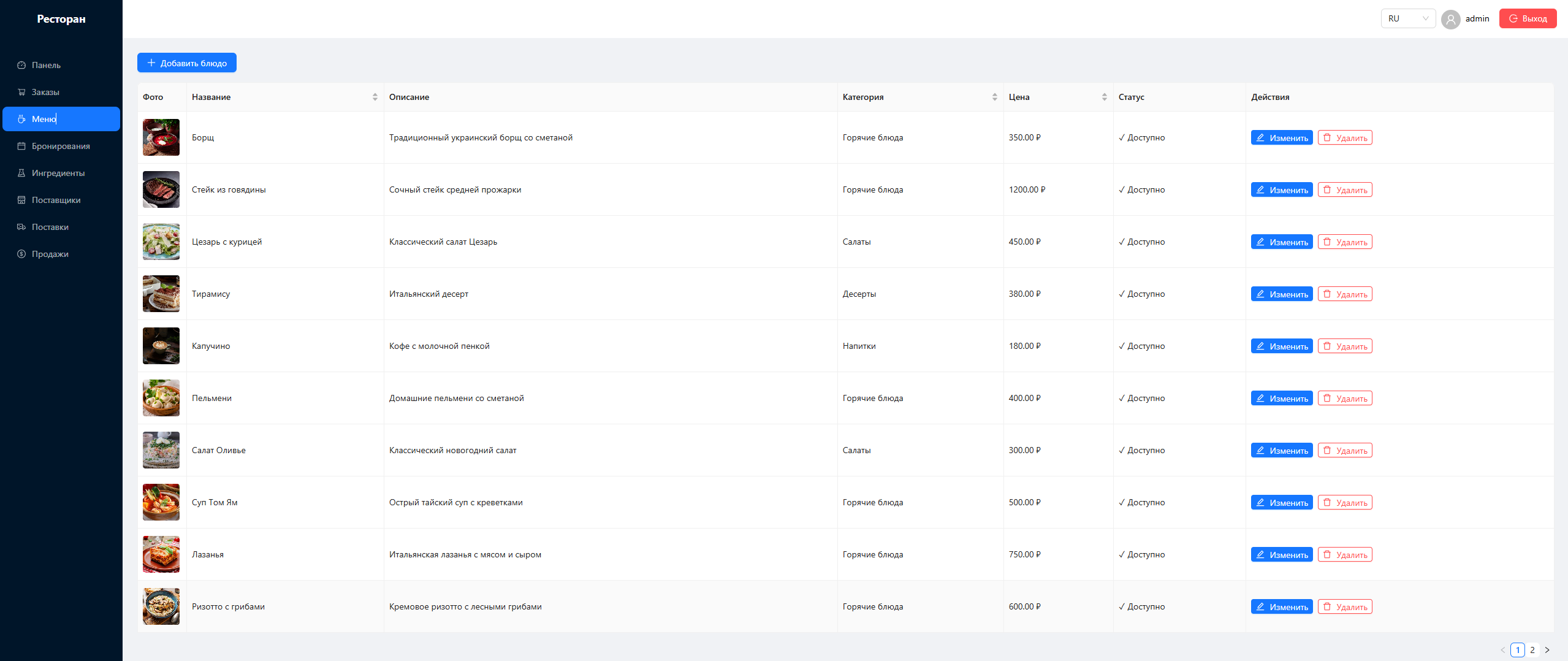
Разработаны Java-классы, соответствующие спроектированной диаграмме классов.

* Сущности (Entities): классы Order, Dish, User и остальные, аннотированные для связи с таблицами БД (ORM Hibernate).
* Репозитории (Repositories): интерфейсы, наследуемые от JpaRepository, обеспечивающие методы save, findById, findAll.
* Сервисы (Services): OrderService: содержит логику оформления заказа. В методе createOrder реализована проверка доступности ингредиентов на складе перед сохранением заказа. В случае нехватки продуктов выбрасывается исключение, и транзакция откатывается.
* Контроллеры (Controllers): Классы, принимающие HTTP-запросы. Например, OrderController принимает POST-запрос на создание заказа, вызывает сервис и возвращает JSON-ответ.

**Реализация уровня представления (Клиентская часть):**

Разработано веб-приложение администратора.

* Реализована форма входа в систему.
* Создана страница управления меню (добавление/редактирование блюд).
* Создана страница просмотра активных заказов с обновлением статусов.

*Рисунок 40 – Интерфейс управления меню*

### 4.5. Тестирование

Заключительным шагом этапа “Создание” является тестирование разработанной системы для подтверждения её работоспособности и соответствия требованиям.

Было проведено функциональное тестирование основных сценариев использования:

1. Сценарий “Авторизация”:

* Действие: Ввод корректного логина и пароля.
* Результат: Система выдает токен доступа и перенаправляет на главную страницу. Тест пройден.

1. Сценарий “Создание заказа”:

* Действие: Добавление блюда в корзину и подтверждение заказа.
* Результат: Заказ сохраняется в БД со статусом «CREATED». Количество ингредиентов на складе уменьшается на соответствующую величину. Тест пройден.

1. Сценарий “Контроль остатков”:

* Действие: Попытка заказать блюдо, для которого не хватает ингредиентов.
* Результат: Система выдает ошибку “Недостаточно ингредиентов на складе”. Заказ не создается. Тест пройден.

Также было проведено автоматизированное тестирование API с использованием скриптов (PowerShell/cURL), которое подтвердило корректную обработку HTTP-запросов и кодов ответов сервера.

# КОНСТРУКТОРСКАЯ ЧАСТЬ

В данном разделе приводится техническое задание на разработку, описывается архитектура системы, схема базы данных и спецификации программных модулей.

Разработанная документация полностью соответствует требованиям Единой системы программной документации (ЕСПД) и может быть использована как для самостоятельной реализации проекта, так и для передачи разработки сторонней команде.

## 1. Техническое задание

Настоящее техническое задание распространяется на разработку информационной системы ресторана, предназначенной для автоматизации процессов обслуживания клиентов и складского учета. Документ определяет требования к функциональным характеристикам, надежности, условиям эксплуатации и составу программной документации. Ниже в тексте представлен титульный лист Технического задания, оформленный в соответствии с требованиями ГОСТ 19.104-78.

### 1.2. Введение

Разрабатываемая система представляет собой программный комплекс для автоматизации деятельности предприятия общественного питания (ресторана). Система предназначена для использования в сфере услуг и управления ресторанным бизнесом.

Актуальность разработки обусловлена необходимостью перехода от бумажного документооборота к цифровым технологиям для повышения конкурентоспособности предприятия на рынке услуг общественного питания.

### 1.3. Основание для разработки

Основанием для разработки является Учебный план направления 09.03.02 «Информационные системы и телекоммуникации», утвержденный Ректором МГТУ им. Н.Э. Баумана, и задание на выполнение научно-исследовательской работы по теме «Информационная система ресторана».

В рамках учебного процесса данная работа направлена на закрепление теоретических знаний и практических навыков в области проектирования информационных систем, полученных при изучении профильных дисциплин. Разработка ведется под контролем научного руководителя и соответствует требованиям федерального государственного образовательного стандарта.

Выполнение проекта осуществляется в сроки, установленные календарным планом на текущий семестр, с обязательным прохождением этапов промежуточного контроля и итоговой защиты результатов.

### 1.4. Требования к функциональным характеристикам

Система должна обеспечивать выполнение следующих функций, определенных на этапе анализа системы:

1. Ведение справочников меню (блюда, ингредиенты) с возможностью добавления, редактирования и удаления записей.
2. Оформление заказов клиентами и сотрудниками с автоматическим расчетом итоговой стоимости.
3. Учет складских остатков в режиме реального времени с автоматическим списанием ингредиентов по технологическим картам.
4. Управление статусами заказов (принят, готовится, готов, оплачен).
5. Аутентификация и авторизация пользователей с разделением прав доступа (Администратор, Менеджер, Официант, Клиент).

### 1.5. Требования к надежности

Для обеспечения надежного функционирования системы предусмотрены следующие меры:

* **Повышение надежности ввода данных:** реализована валидация входных форм (проверка типов данных, обязательных полей, диапазона значений). Используются маски ввода для дат и телефонных номеров.
* **Защита информации от разрушения:** применяется механизм транзакций СУБД (ACID), гарантирующий атомарность операций (заказ создается только при успешном списании всех ингредиентов). В случае сбоя происходит автоматический откат изменений (ROLLBACK).
* **Защита от несанкционированного доступа:** доступ к административным функциям закрыт механизмом аутентификации на основе JWT-токенов. Пароли хранятся в хешированном виде.

### 1.6. Условия эксплуатации

* Условия запуска: для работы серверной части необходима установленная среда Java Runtime Environment (JRE) версии 17 и выше, а также доступ к СУБД PostgreSQL.
* Режим работы: система рассчитана на круглосуточную непрерывную работу (24/7).
* Требования к персоналу: систему может обслуживать пользователь-непрограммист, знакомый с основными принципами работы в среде операционной системы Windows и веб-браузере. Система снабжена дружественным интерфейсом, всплывающими подсказками и уведомлениями об ошибках.

### 1.7. Требования к составу и параметрам технических средств

Для функционирования системы необходим следующий состав технических средств:

* Сервер: процессор с частотой от 2 ГГц, ОЗУ от 4 ГБ, дисковое пространство от 20 ГБ.
* Клиентские рабочие места: персональные компьютеры или планшеты с современным веб-браузером (Chrome, Firefox, Safari), разрешение экрана не менее 1024x768.
* Сетевое оборудование: локальная сеть или доступ в Интернет с пропускной способностью не менее 10 Мбит/с.

### 1.8. Требования к информационной и программной совместимости

Для обеспечения корректного взаимодействия компонентов системы, а также её интеграции с внешним программным окружением, устанавливаются следующие требования к совместимости. Соблюдение данных требований гарантирует кроссплатформенность серверной части, унификацию протоколов обмена данными и стабильную работу системы в гетерогенной сетевой среде:

* Программная среда: операционная система Windows/Linux/macOS;
* Протоколы интерфейсов: взаимодействие клиента и сервера осуществляется по протоколу HTTP/HTTPS (REST API). Формат обмена данными — JSON;
* СУБД: PostgreSQL версии 14 и выше;
* Система должна обеспечивать корректную работу при одновременном доступе не менее 50 пользователей.

### 1.9. Требования к маркировке и упаковке

В связи с тем, что программное изделие представляет собой нематериальный актив и передается заказчику посредством цифровых каналов связи, требования к физической маркировке и таре не предъявляются. Идентификация версии программного обеспечения осуществляется через метаданные в исполняемых файлах и записях в системе контроля версий.

Соответственно, требований нет.

### 1.10. Требования к транспортировке и хранению

Программный продукт хранится в репозитории системы контроля версий (Git) и может распространяться в виде Docker-образов или JAR-архива.

При передаче программного изделия через открытые каналы связи (Интернет) должны применяться средства криптографической защиты (HTTPS, SSH) для обеспечения конфиденциальности и целостности передаваемой информации.

Для долгосрочного хранения исходного кода и документации рекомендуется использовать надежные облачные сервисы (например, GitHub или GitLab) с обязательным созданием локальных резервных копий на жестких дисках. Это защитит проект от потери данных в случае сбоя удаленного сервера или блокировки аккаунта.

### 1.11. Требования к программной документации

Разрабатываемая система должна сопровождаться комплектом программной документации, достаточным для её установки, эксплуатации и последующего сопровождения. Состав документации определяется в соответствии с требованиями Единой системы программной документации (ЕСПД), в частности ГОСТ 19.101–77 [6] и ГОСТ 19.201–78 [8].

В комплект поставки должны входить следующие документы:

1. Текст программы (ГОСТ 19.401): Исходный код должен быть снабжен подробными комментариями, поясняющими назначение основных классов, методов и нетривиальных алгоритмов.
2. Руководство системного программиста (ГОСТ 19.501): Документ должен содержать пошаговые инструкции по развертыванию серверной части (Back-end) и клиентского приложения (Front-end), настройке подключения к СУБД PostgreSQL и конфигурации переменных окружения. Данный документ допускается представить в формате README.md в корневом каталоге репозитория.
3. Руководство оператора (ГОСТ 19.505): Описание пользовательского интерфейса и сценариев работы основных групп пользователей (администраторов, менеджеров, официантов).

Вся документация должна быть выполнена на русском языке и предоставлена в электронном виде. В состав документации также должно входить описание процедур резервного копирования базы данных (Backup) и восстановления системы после сбоев (Recovery).

### 1.12. Технико-экономические показатели

Ожидаемая экономическая эффективность внедрения системы обусловлена снижением операционных расходов. Автоматизация рутинных операций позволит сократить время обслуживания одного клиента на 20%, что увеличит пропускную способность зала в часы пик. Кроме того, внедрение строгого складского учета снизит потери продуктов от порчи и хищений минимум на 15%, что обеспечит окупаемость разработки в течение первого года эксплуатации.

### 1.13. Стадии и этапы разработки

Таблица 11 – План-график выполнения работ

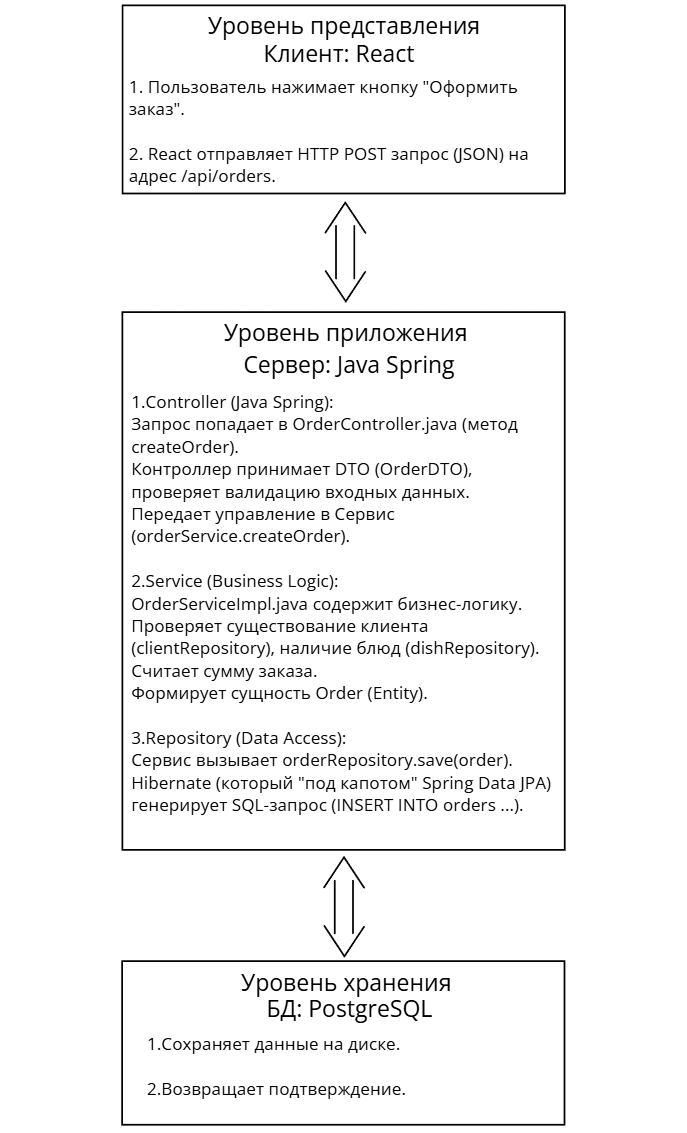
| № этапа | Содержание этапа | Срок окончания | Отчетные материалы |
| --- | --- | --- | --- |
| 1 | Анализ предметной области | 15.09.2025 | Раздел «Анализ» в РПЗ |
| 2 | Проектирование системы и БД | 15.10.2025 | Раздел «Проектирование» |
| 3 | Программная реализация | 15.11.2025 | Исходный код, Глава «Создание» |
| 4 | Тестирование и отладка | 02.12.2025 | Протоколы тестирования |

По завершении каждого этапа производится контроль качества выполненных работ и согласование результатов с научным руководителем. Сроки могут корректироваться по согласованию сторон при возникновении непредвиденных технических сложностей.

## 2. Архитектура системы

В соответствии с требованиями методических указаний для разработки информационной системы была выбрана трехуровневая архитектура программного обеспечения.

На рисунке 41 представлена общая схема архитектуры с указанием потоков данных для оформления заказа официантом.

**

*Рисунок 41 – Трехуровневая архитектура системы на примере оформления заказа официантом*

Назначение уровней:

1. **Уровень представления (Frontend):** Реализован как SPA (Single Page Application) на базе библиотеки React.

Обеспечивает интерфейс с пользователем, валидацию ввода и отображение данных.

1. **Уровень приложения (Backend):** Реализован на Java (Spring Boot).

Содержит бизнес-логику, обрабатывает запросы от клиента, управляет транзакциями и взаимодействует с базой данных.

1. **Уровень хранения данных (Database):** Реализован на СУБД PostgreSQL.

Отвечает за хранение, выборку и целостность данных.

## 3. База данных

### 3.1. Обоснование выбора СУБД

Для реализации уровня хранения данных выбрана СУБД PostgreSQL. Выбор обусловлен следующими факторами:

* Поддержка стандарта SQL и реляционной модели данных [4].
* Наличие механизмов обеспечения целостности (FOREIGN KEY, CHECK, UNIQUE), необходимых для реализации нормализованной схемы (4НФ).
* Высокая производительность и надежность (ACID).
* Открытый исходный код и активная поддержка сообществом.

Выбранная СУБД полностью удовлетворяет требованиям к надежности хранения данных и скорости обработки транзакций, характерных для многопользовательских систем.

### 3.2. Схема базы данных и описание таблиц

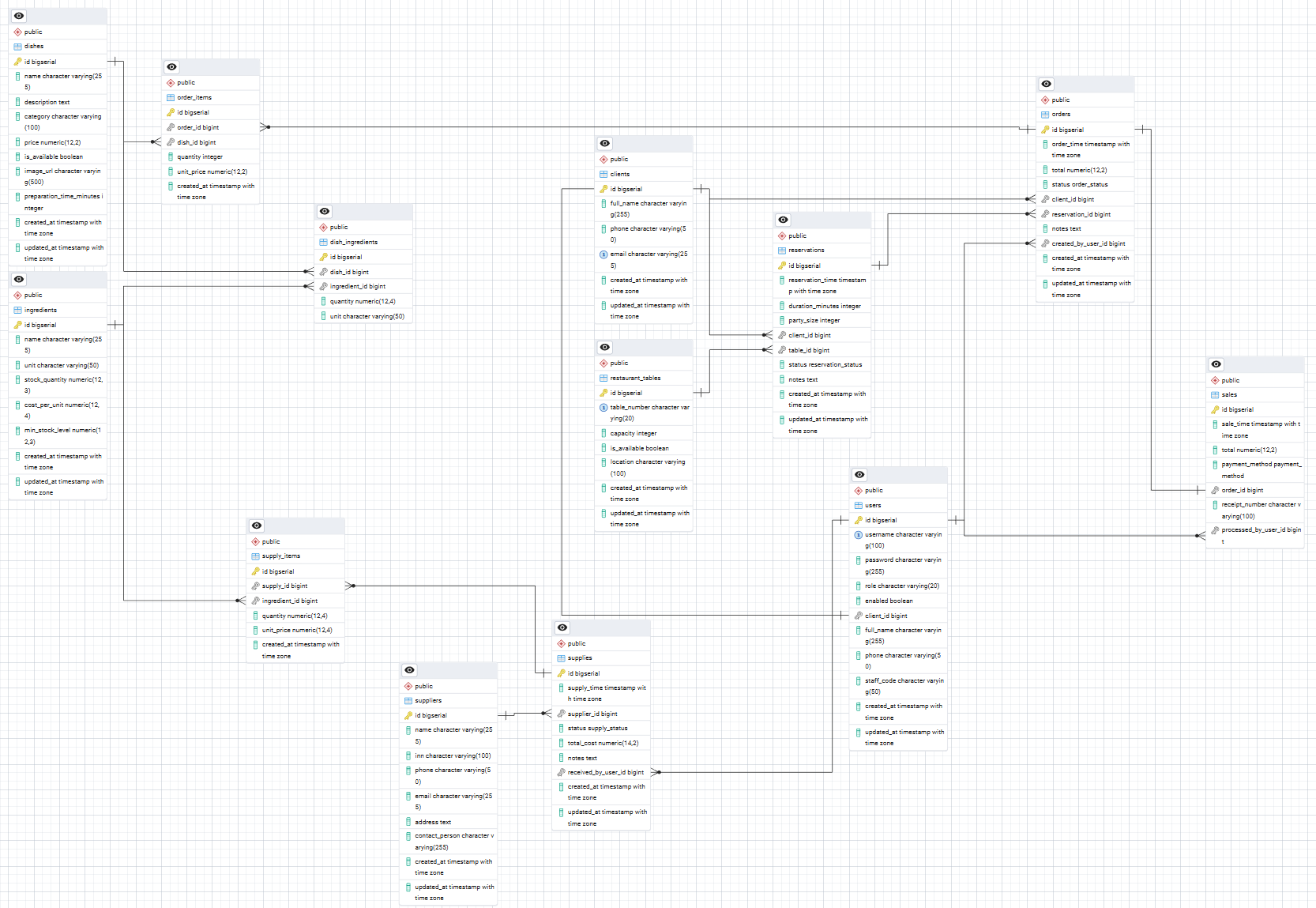
Физическая модель базы данных, разработанная на этапе проектирования, была реализована с помощью SQL-скриптов. В схеме активно используются перечисления (ENUM) для типизации статусов и ролей.

Основные таблицы системы:

1. **users** – таблица аутентификации пользователей. Содержит учетные данные (логин, хеш пароля), роль (user\_role: ADMIN, MANAGER, WAITER, CLIENT) и ссылки на профиль клиента или данные сотрудника;
2. **clients** – расширенные профили гостей ресторана (ФИО, телефон, email), которые могут быть привязаны к пользователям;
3. **restaurant\_tables** – справочник физических столиков в ресторане (номер, вместимость, расположение);
4. **reservations** – журнал бронирования столиков. Связывает клиента и столик на определенное время, содержит статус брони (reservation\_status);
5. **ingredients** – складской учет продуктов (название, единица измерения, текущий остаток, минимальный порог запаса);
6. **dishes** – меню ресторана (название, цена, категория, время приготовления, признак доступности);
7. **dish\_ingredients** – таблица связи «многие-ко-многим», определяющая рецептуру блюд (состав и граммовки ингредиентов для списания со склада);
8. **orders** – заголовки заказов. Содержат итоговую сумму, текущий статус (order\_status), ссылку на бронирование или клиента, а также сотрудника, создавшего заказ;
9. **order\_items** – детализация заказа. Хранит список заказанных блюд, их количество и фиксированную цену на момент заказа.
10. **sales** – реестр финансовых операций. Фиксирует факт оплаты заказа, метод оплаты и чек;
11. **suppliers** – справочник поставщиков продуктов;
12. **supplies** и **supply\_items** – учет поставок ингредиентов на склад (заголовок накладной и перечень товаров).

Для обеспечения целостности данных на уровне СУБД реализованы:

* Внешние ключи (FOREIGN KEY) с правилами каскадного обновления или удаления (ON DELETE CASCADE / SET NULL);
* Проверки ограничений (CHECK), запрещающие отрицательные значения цены, количества товара (quantity > 0) и вместимости столов;
* Уникальные индексы (UNIQUE) для предотвращения дублирования логинов, email и номеров столов.



*Рисунок 42 – Схема реализованной базы данных*

## 4. Серверная часть системы

### 4.1. Обоснование средств разработки

Серверная часть реализована на языке Java с использованием фреймворка Spring Boot.

* Java: Строго типизированный объектно-ориентированный язык, обеспечивающий надежность и удобство поддержки кода.
* Spring Boot: Фреймворк, предоставляющий готовые компоненты для веб-разработки (Spring MVC), безопасности (Spring Security) и работы с БД (Spring Data JPA), что значительно ускоряет процесс создания системы.

Сочетание языка Java и фреймворка Spring Boot является отраслевым стандартом для создания корпоративных систем, гарантируя высокую производительность, безопасность и легкость в поддержке кода.

### 4.2. Описание схемы бизнес-логики

Логика работы системы построена по модульному принципу. Приложение разбито на слои:

1. Controllers (Контроллеры): Принимают HTTP-запросы, вызывают сервисы и возвращают ответы.
2. Services (Сервисы): Реализуют бизнес-правила (расчет стоимости, проверка остатков).
3. Repositories (Репозитории): Выполняют операции с базой данных.

**

*Рисунок 43 – Структура серверной части, в которой реализована бизнес-логика*

Данная схема обеспечивает четкое разделение ответственности. Контроллеры занимаются только валидацией входных данных и формированием ответа, сервисы содержат чистую бизнес-логику, а репозитории инкапсулируют логику доступа к данным, что упрощает модульное тестирование каждого слоя.

### 4.3. Спецификация модулей

В системе выделены следующие основные модули: AuthModule (авторизация), MenuModule (управление меню), OrderModule (заказы), WarehouseModule (склад). Ниже приведены спецификации для ключевых модулей.

**Форма 1. Спецификация на модуль OrderService (Управление заказами)**

1. Системное имя модуля: OrderService
2. Функции модуля:

* Создание нового заказа с проверкой доступности ингредиентов.
* Изменение статуса заказа (Создан – Готовится – Готов).
* Расчет полной стоимости заказа.
* Алгоритм: при создании заказа для каждого блюда извлекается список ингредиентов, проверяется их количество на складе. Если req\_qty > stock\_qty, выбрасывается исключение. Иначе происходит уменьшение остатков.

1. Входные данные:

* Объект CreateOrderDTO: список ID блюд и их количества, номер столика.

1. Выходные данные:

* Объект Order: созданная сущность заказа с присвоенным ID и временной меткой.
* Таблица products: обновленные значения поля quantity.

1. Внешние эффекты:

* При успехе: возврат HTTP 200 OK и JSON с данными заказа.
* При ошибке: возврат HTTP 400 Bad Request с сообщением "Недостаточно товара X на складе".

1. Способ вызова: вызывается из OrderController при получении POST-запроса /api/orders.
2. Список вызываемых модулей:

 ProductRepository, OrderRepository, DishRepository.

**Форма 1. Спецификация на модуль WarehouseService (Склад)**

1. Системное имя модуля: WarehouseService
2. Функции модуля:

* Получение текущих остатков.
* Регистрация поставки (увеличение остатков).
* Списание продуктов.

1. Входные данные: ID продукта, количество изменения.
2. Выходные данные: обновленная запись товара в таблице products.
3. Внешние эффекты: сообщения об ошибках при попытке списать больше, чем есть в наличии.
4. Способ вызова: вызывается из ProductController или OrderService.

## 5. Интерфейс пользователя

Интерфейс системы реализован в виде двух независимых веб-приложений: клиентского приложения (для гостей ресторана) и панели администратора (для сотрудников: администраторов, менеджеров и официантов). Это обеспечивает разделение доступа и удобство использования для разных групп пользователей.

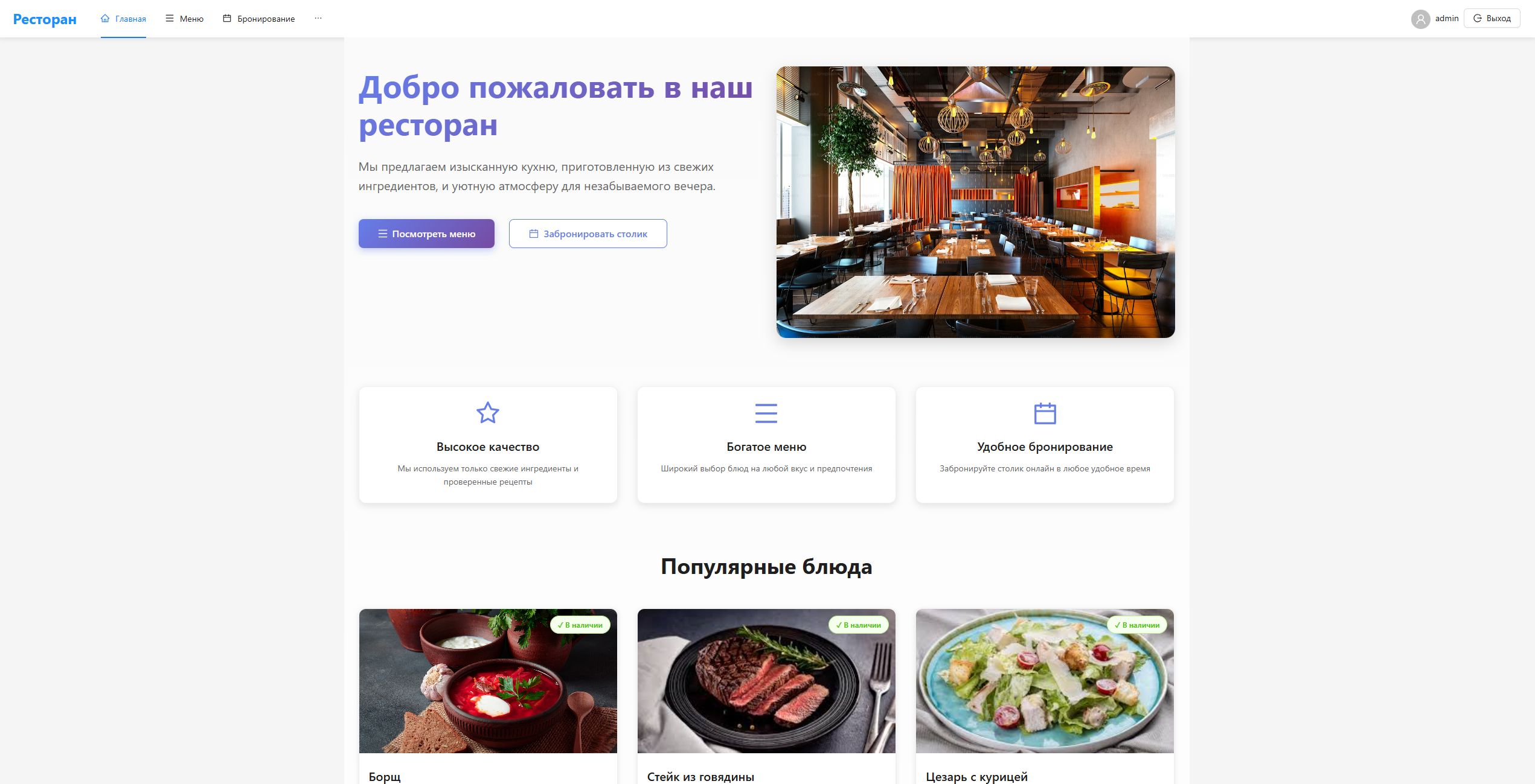
Проектирование интерфейса выполнялось с учетом принципов юзабилити и доступности. Основной акцент сделан на интуитивность управления, чтобы минимизировать время обучения персонала работе с новой системой. Цветовая гамма и шрифты подобраны таким образом, чтобы снизить нагрузку на зрение при длительной работе.

### 5.1. Клиентское приложение

Предназначено для гостей ресторана. Позволяет просматривать меню, регистрироваться и бронировать столики онлайн.

#### Главная страница и Меню

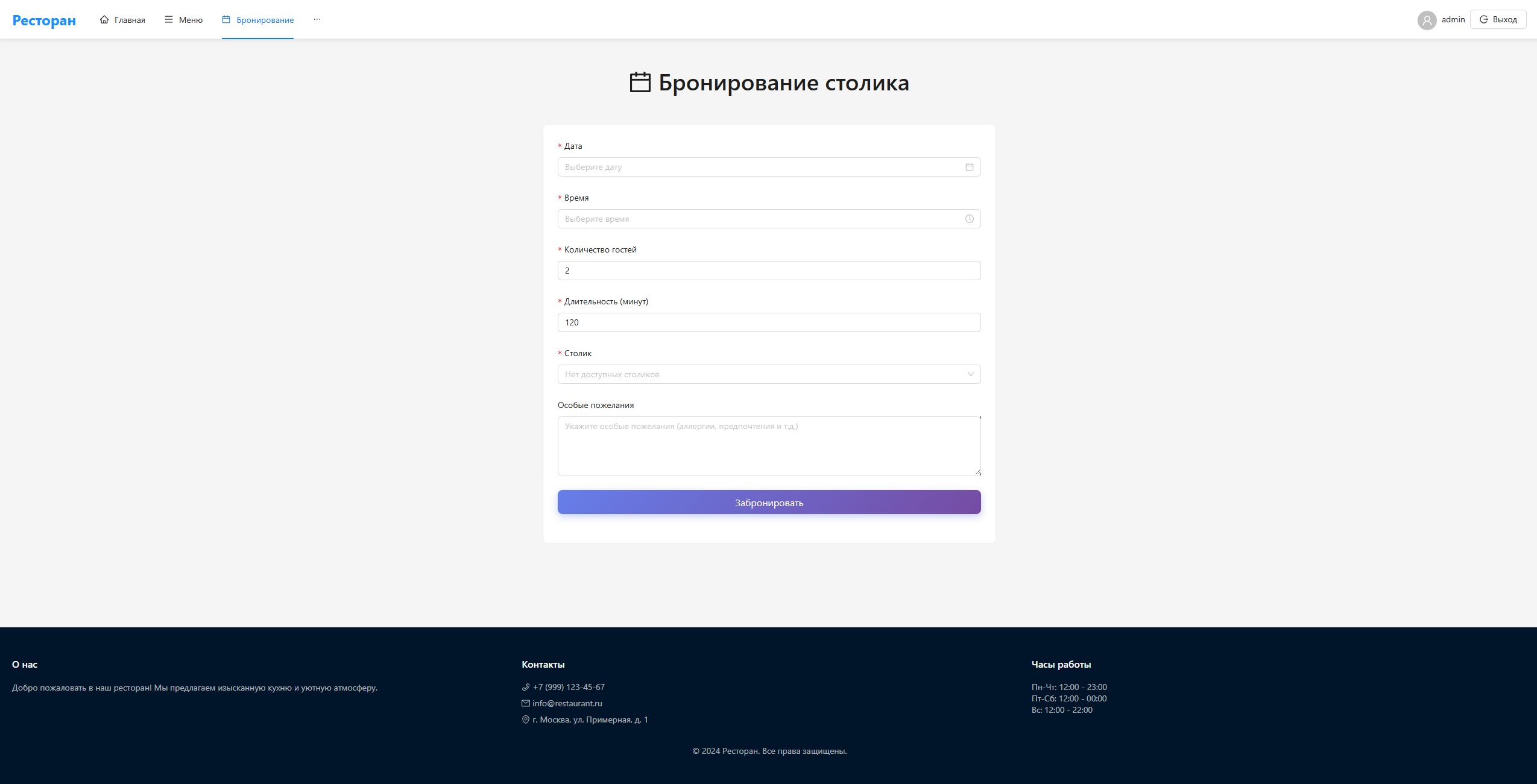
Публичная часть сайта. Содержит красочное представление меню с фотографиями блюд, ценами и описанием. Пользователь может фильтровать блюда по категориям.



*Рисунок 44 – Главная страница клиентского приложения*

#### Личный кабинет и Бронирование

Доступен после авторизации/регистрации. Позволяет клиенту выбрать дату, время и количество персон для бронирования столика. В личном кабинете отображается история бронирований и их текущий статус.



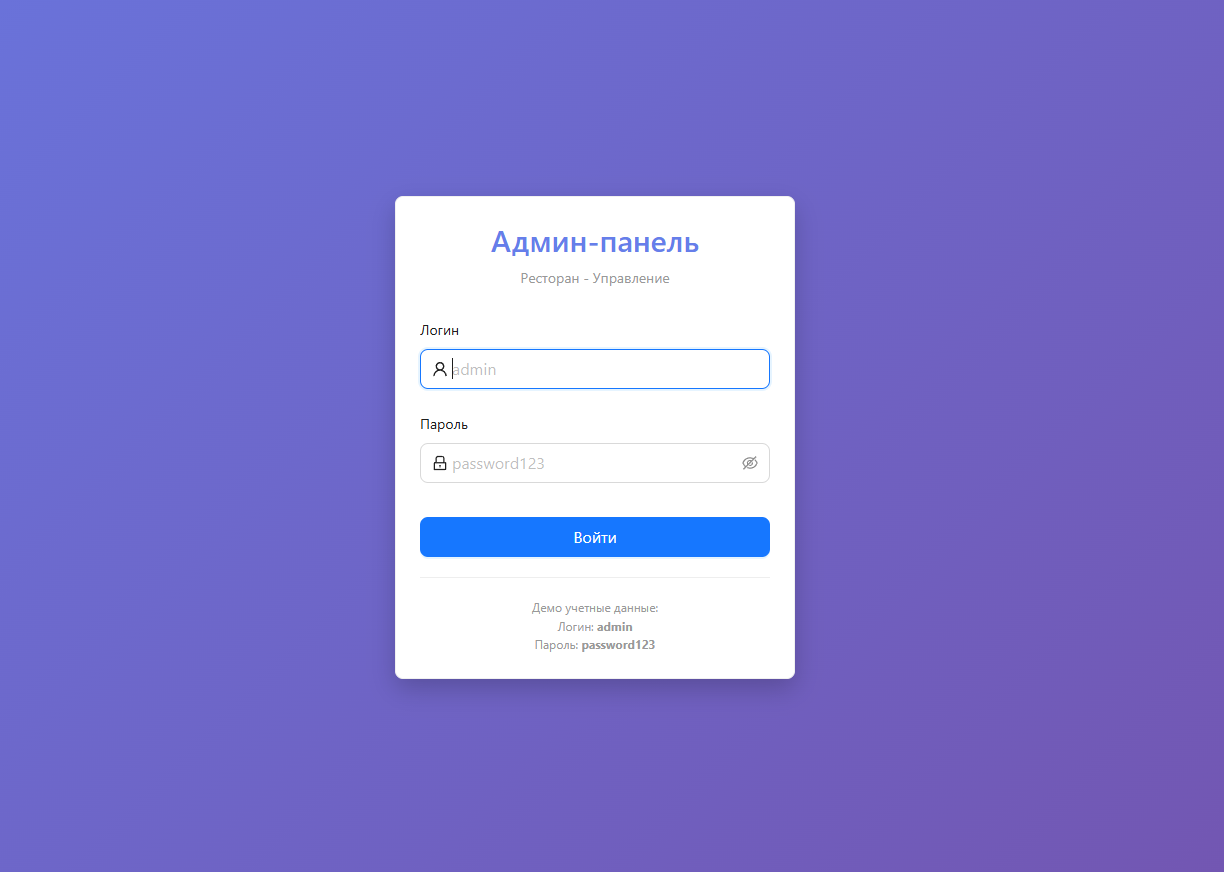
*Рисунок 45 – Страница бронирований клиентского приложения*

### 5.2. Приложение для персонала

Закрытая часть системы, предназначенная для управления рестораном. Доступ разграничивается на основе ролей (ADMIN, MANAGER, WAITER).

#### Экран авторизации

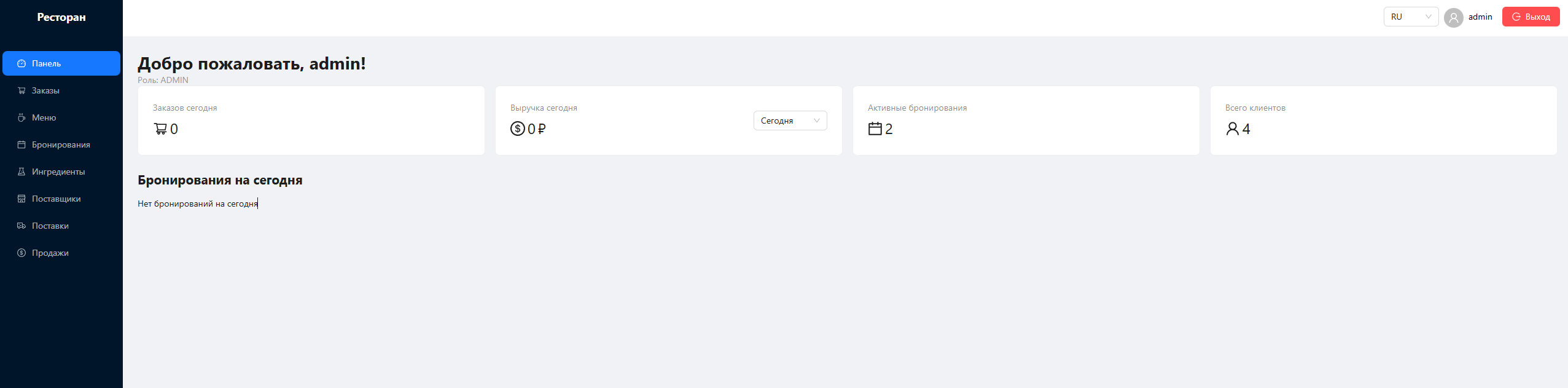
Единая точка входа для сотрудников. Содержит поля «Логин», «Пароль». После успешного входа система перенаправляет пользователя на соответствующий его роли раздел.



*Рисунок 46 – Страница входа приложения для персонала*

#### Dashboard (Сводная панель)

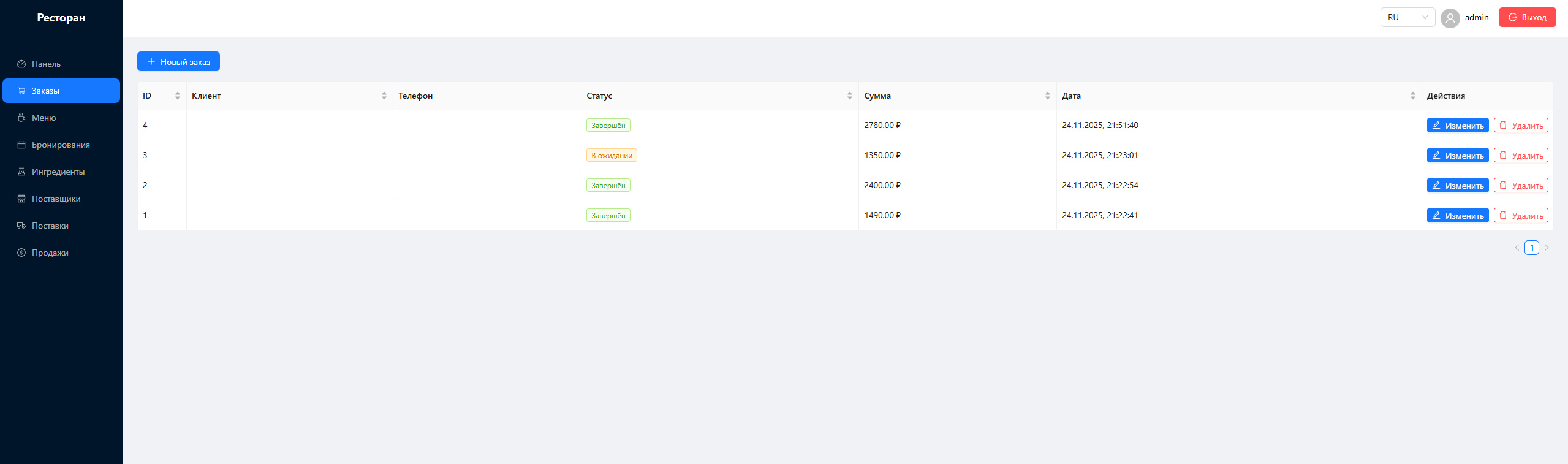
Главный экран менеджера/администратора. Отображает ключевые метрики в реальном времени: выручку за сегодня, количество выполненных заказов, активные бронирования и количество обслуженных клиентов.



*Рисунок 47 – Страница сводной панели приложения для персонала*

#### Управление заказами (Интерфейс официанта)

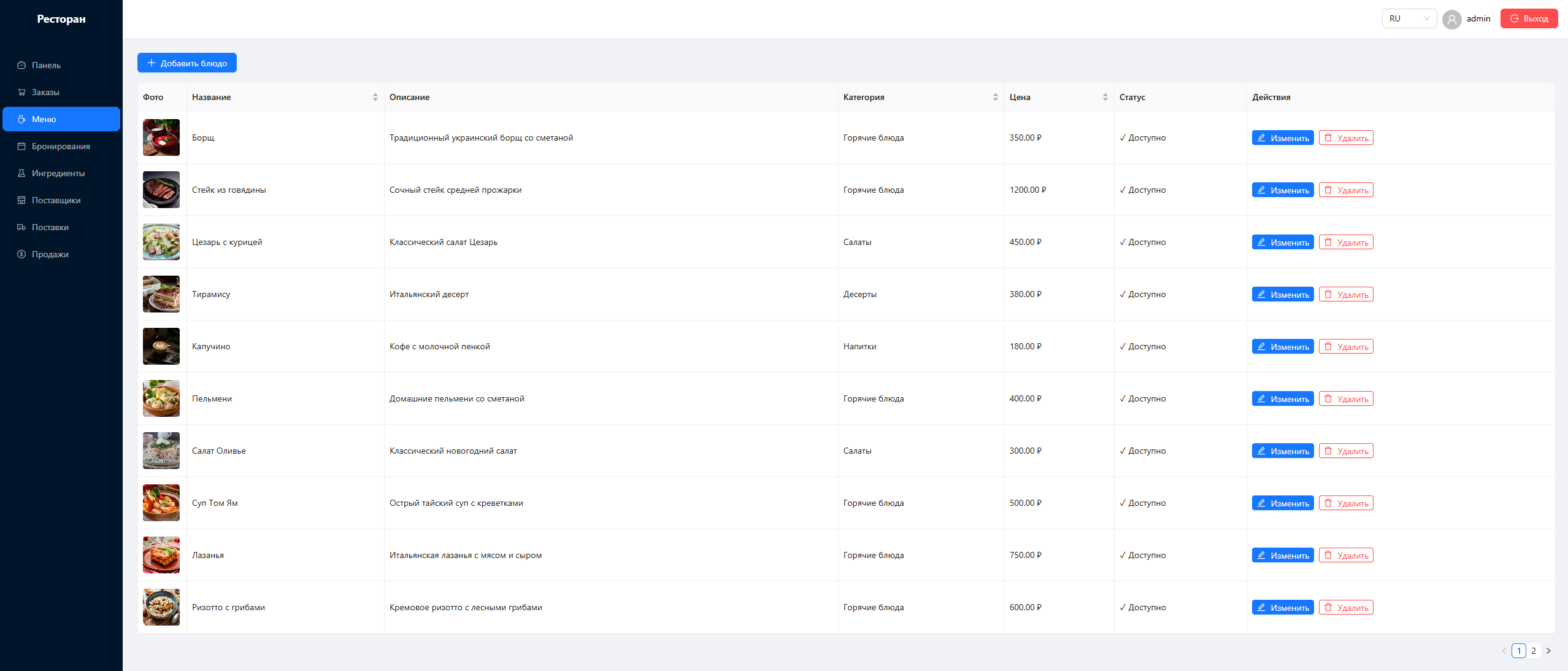
Рабочее место официанта. Позволяет создавать новые заказы для столиков, добавлять блюда в заказ, менять статус заказа (“В ожидании”, “Готовится”, “Завершён”, “Отменён”) и отправлять его на кухню.



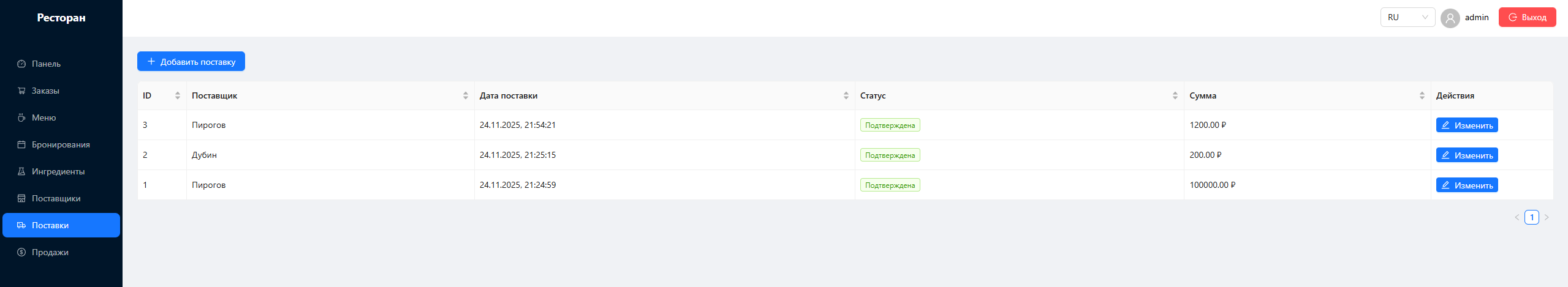
*Рисунок 48 – Страница управления заказами приложения для персонала*

#### Склад и Управление меню

Интерфейс для менеджера. Позволяет редактировать технологические карты блюд, следить за остатками ингредиентов на складе, оформлять новые поставки и управлять списком поставщиков.



*Рисунок 49 – Страница управления меню приложения для персонала*



*Рисунок 50 – Страница контроля поставок приложения для персонала*

# ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

В данном разделе описывается процесс исследования и тестирования разработанной информационной системы. Основной целью является проверка оптимальности решения, производительности, отказоустойчивости, а также выполнение системного и интеграционного тестирования компонентов.

Комплексный подход к тестированию, описанный в данном разделе, гарантирует, что конечный продукт будет работать стабильно и предсказуемо в реальных условиях эксплуатации, минимизируя риски сбоев при обслуживании клиентов.

## 1. Анализ исходных данных

После реализации и отладки системы была проведена процедура комплексного исследования. Для выявления скрытых ошибок и недоработок были подготовлены специальные наборы тестовых данных, охватывающие различные режимы функционирования.

В ходе подготовки к тестированию были проверены:

1. Выполнение всех заявленных функций (CRUD-операции, расчеты).
2. Правильность настроек системы и прав доступа.
3. Надежность и целостность данных при транзакциях.
4. Работа системы в трех областях данных:

* Нормальные данные: Стандартные заказы, корректные цены.
* Экстремальные (граничные) данные: Нулевые цены, пустые строки, максимально допустимые значения полей.
* Начальные данные: Работа с пустой базой данных (первый запуск).

Реакция системы на неверные данные определена как выдача понятного сообщения об ошибке без прекращения работы сервера (HTTP коды 400/404).

Результаты исследования подытожены в таблице 5.1.

Таблица 12 – Анализ исходных данных и проверяемых режимов

| Набор исходных данных | Проверяемый режим, функция, алгоритм |
| --- | --- |
| Нормальный: ID блюда=1, Кол-во=2 (при остатке 100) | Функция создания заказа. Проверяется алгоритм списания продуктов и сохранения записи в БД. |
| Экстремальный: Цена блюда = 0.01 руб. | Валидация формы. Проверяется корректность работы с дробными числами и минимальными ценами. |
| Ошибочный: ID блюда=9999 (не существует) | Обработка исключений. Проверяется алгоритм поиска в БД и возврат ошибки "EntityNotFound". |
| Ошибочный: Кол-во заказа = Остаток на складе + 1 | Контроль целостности. Проверяется запрет на создание заказа при нехватке товара (транзакция должна откатиться). |
| Пустой: Логин="", Пароль="" | Авторизация. Проверяется защита от входа с пустыми учетными данными. |

## 2. Тестирующие драйверы и заглушки

Тестирование системы проводилось комбинированным методом. Так как разработка велась по методологии “Снизу-вверх” (от СУБД к серверу, от сервера к клиенту) для проверки серверной части использовались тестирующие драйверы.

Выбор конкретных инструментов для создания драйверов и заглушек был обусловлен технологическим стеком проекта. Поскольку серверная часть написана на Java с использованием Spring Boot, наиболее эффективным решением стало использование встроенных средств тестирования этого фреймворка, которые позволяют легко подменять реальные компоненты на тестовые имитации.

Использование такого подхода позволило выявить и исправить ошибки на ранних стадиях, еще до того, как был полностью готов пользовательский интерфейс. Это существенно сократило время общей отладки системы.

### Тестирующий драйвер (Driver)

В роли тестирующего драйвера использовался инструмент Swagger UI, который является удобным инструментом для тестирования и документации.

Драйвер выполнял следующие функции:

1. Вызывал тестируемый модуль (например, OrderController.createOrder()).
2. Передавал подготовленный JSON-объект с исходными данными (например, { "dishId": 1, "quantity": 5 }).
3. Принимал полученный HTTP-ответ (код статуса и тело ответа).
4. Позволял визуально сравнить полученный результат с ожидаемым.

Также были разработаны скрипты автоматизации на PowerShell (например, smoke\_reservation.ps1), которые выступали в роли автоматического драйвера, последовательно вызывающего цепочку модулей:

*Авторизация – Получение меню – Создание заказа*.

Кроме того, использование Swagger UI в качестве драйвера позволяет наглядно продемонстрировать работу API без необходимости развертывания клиентской части приложения.

Скрипты на PowerShell также выполняли функцию нагрузочного тестирования в упрощенном режиме. Запуская их в цикле, можно было проверить, как система справляется с многократным созданием заказов за короткий промежуток времени.

### Тестирующие заглушки

При модульном тестировании отдельных сервисов (Unit Testing) использовались заглушки для имитации работы базы данных. С помощью библиотеки Mockito создавались объекты-заглушки (Mocks) для репозиториев (ProductRepository), которые вместо реального обращения к БД возвращали заранее подготовленные объекты. Это позволило проверить логику расчетов в отрыве от состояния реальной базы данных.

Еще одним важным преимуществом использования заглушек стала скорость выполнения тестов. Обращение к реальной базе данных занимает время, а заглушка срабатывает мгновенно. Благодаря этому полный набор из сотен модульных тестов выполняется всего за несколько секунд, что позволяет запускать их автоматически при каждом сохранении кода.

Также заглушки позволили протестировать редкие и аварийные ситуации, которые сложно воспроизвести в реальности. Например, с их помощью была сымитирована ситуация разрыва соединения с базой данных во время транзакции, чтобы проверить, корректно ли система откатывает изменения.

## 3. Протокол тестирования

Результаты проведенного тестирования оформлены в виде протокола.

Таблица 13 – Протокол тестирования

|  |  |  |
| --- | --- | --- |
| Наборы исходных данных | Ожидаемый результат | Полученный результат |
| Тест №1. Вход в систему.  Логин: admin  Пароль: 1234 | Успешная авторизация. Получен JWT-токен. HTTP статус 200. | Успешная авторизация. Токен получен. HTTP 200. |
| Тест №2. Заказ доступного товара. Товар ID=1 (остаток 10), кол-во=2. | Заказ создан. Новый остаток=8. HTTP статус 201. | Заказ создан. В БД остаток 8. HTTP 201. |
| Тест №3. Заказ дефицитного товара.  Товар ID=1 (остаток 8), кол-во=10. | Ошибка "Недостаточно товара". Заказ не создан. Остаток=8. HTTP 400. | Ошибка получена. Остаток не изменился (8). HTTP 400. |
| Тест №4. SQL-инъекция.  В поле поиска: ' OR '1'='1 | Система экранирует ввод, поиск не возвращает лишних данных. | Пустой результат поиска (или корректная ошибка). Данные не утекли. |

*Проектант: Измайлов А.Р. \_\_\_\_\_\_\_\_\_\_\_\_\_\_*

## 4. Исследование системы на производительность и отказоустойчивость

В рамках исследования были проанализированы характеристики производительности и надежности системы.

**Отказоустойчивость:**

1. Для обеспечения целостности данных использован механизм транзакций СУБД PostgreSQL уровня READ COMMITTED. Это гарантирует, что параллельные запросы на создание заказов корректно уменьшают остатки на складе, исключая ситуацию “Race Condition” (гонки данных).
2. Реализована глобальная обработка исключений (GlobalExceptionHandler), которая перехватывает системные ошибки (например, разрыв связи с БД) и возвращает клиенту понятное сообщение в формате JSON, предотвращая “падение” сервера.

**Производительность:**

Для оптимизации запросов и повышения скорости передачи информации были реализованы следующие решения:

* Пагинация данных: Списки заказов и истории операций загружаются страницами (по 10-20 записей), что снижает нагрузку на канал связи и оперативную память браузера;
* Индексация БД: Созданы B-Tree индексы для полей внешних ключей (order\_id, dish\_id), что ускорило выполнение запросов JOIN при формировании отчетов;
* DTO (Data Transfer Objects): Сервер передает клиенту только необходимые данные (без лишних служебных полей сущностей), минимизируя объем трафика.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения научно-исследовательской работы была разработана информационная система, предназначенная для автоматизации ключевых процессов работы сети ресторанов. Проект охватил полный цикл разработки программного обеспечения: от анализа предметной области до реализации и тестирования готового продукта.

Основные результаты работы включают:

* проведен анализ предметной области и сформированы функциональные требования к системе;
* разработаны диаграммы вариантов использования, последовательности и классов, отражающие архитектуру системы;
* спроектирована и нормализована до четвёртой нормальной формы (4НФ) схема базы данных;
* реализована физическая база данных в СУБД PostgreSQL;
* разработано программное обеспечение на базе трехуровневой архитектуры с использованием технологий Java (Spring Boot) и React;
* проведено комплексное тестирование системы, подтвердившее её работоспособность, надежность и соответствие техническому заданию.

Практическая ценность выполненной работы заключается в создании полностью функционального программного комплекса, обеспечивающего:

* автоматизацию оформления заказов и передачи их на кухню;
* управление меню и учет складских остатков в реальном времени;
* минимизацию ошибок персонала за счет валидации данных и автоматических расчетов;
* повышение скорости обслуживания клиентов.

Пояснительная записка и графические материалы оформлены в соответствии с действующими стандартами [5]. Разработанная система готова к опытной эксплуатации и может быть внедрена в реальных условиях предприятия общественного питания с возможностью дальнейшего масштабирования.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Фаулер, М. UML. Основы / М. Фаулер. – 4-е изд. – Санкт-Петербург: Питер, 2020. – 256 с. – ISBN 978-5-4461-1109-3. – Текст: непосредственный.
2. Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, А. Джекобсон. – 2-е изд. — Москва: Вильямс, 2021. – 304 с. – Текст: непосредственный.
3. Аммерман, Д. Современное проектирование баз данных / Д. Аммерман. – Москва: Диалектика, 2020. – 432 с. – Текст: непосредственный.
4. Иванов, И. И. Реляционные базы данных: учебное пособие / И. И. Иванов. – Москва: ИНФРА-М, 2019. – 272 с. – Текст: непосредственный.
5. ГОСТ 34.320–2017. Информационные технологии. Требования к оформлению документов. – Введ. 2019–01–01. – Москва: Стандартинформ, 2019 – Текст: непосредственный.
6. ГОСТ 19.101–77. Единая система программной документации. Виды программ и программных документов. – Введ. 1980–01–01. – Москва: Стандартинформ, 2010. – Текст: непосредственный.
7. ГОСТ 19.201–78. Единая система программной документации. Текст программ. Требования к содержанию и оформлению. – Введ. 1980–01–01. – Москва: Стандартинформ, 2010. – Текст: непосредственный.
8. ГОСТ 19.204–78. Единая система программной документации. Описание программы. – Введ. 1980–01–01. – Москва: Стандартинформ, 2010. – Текст: непосредственный.
9. PlantUML: официальная документация [Электронный ресурс]. – URL: https://plantuml.com/ru (дата обращения: 02.12.2025).
10. Онлайн-редактор диаграмм Draw.io [Электронный ресурс]. – URL: https://drawio.app (дата обращения: 02.12.2025).
11. Инструмент для построения ER-диаграмм Dbdiagram.io [Электронный ресурс]. – URL: https://dbdiagram.io (дата обращения: 02.12.2025).

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | УТВЕРЖДАЮ  Проректор НУК ИУ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Пролетарский  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 202\_ г. | СОГЛАСОВАНО  Заведующий кафедрой ИУ3 “Информационные системы и телекоммуникации”  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Н. Алфимцев  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 202\_ г. | |

**Техническое задание**

на разработку

научно-исследовательской работы (НИР)

«Информационная система ресторана»

Исполнитель: Измайлов А. Р.\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель: Боровик И. Г. \_\_\_\_\_\_\_\_\_\_\_\_\_

Москва

2025